Etude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice

Chi Thanh Nguyen

THÈSE EN CO-TUTELLE

Présentée par

NGUYỄN Chí Thành

Pour obtenir les titres de

Docteur de l'Université Joseph Fourier, Grenoble et Docteur du Viêt-nam

Spécialité : DIDACTIQUE DES MATHEMATIQUES

ÉTUDE DIDACTIQUE DE L'INTRODUCTION D'ÉLÉMENTS D'ALGORITHMIQUE ET DE PROGRAMMATION DANS L'ENSEIGNEMENT MATHÉMATIQUE SECONDAIRE À L'AIDE DE LA CALCULATRICE

Soutenue le 16 Décembre 2005

Composition du jury :

Annie **BESSOT**	Co-directrice de thèse
Philippe **JORRAND**	Examinateur
Jean-Baptiste **LAGRANGE**	Rapporteur
LÊ THỊ Hoài Châu	Examinatrice
LÊ VĂN Tiến	Rapporteur
NGUYỄN BÁ Kim	Co-directeur de thèse
Alain **BIREBENT**	Invité

Thèse préparée au sein des équipes

Didactique des Mathématiques (DDM), Laboratoire Leibniz – IMAG

Formation de Troisième cycle, École normale supérieure de Hanoï

Abstract

There is a fundamental solidarity between mathematics and computer science that is based on the history and the current practice of these two disciplines. A proof of this is constant resort to algorithms in the resolutions of fundamental mathematical problems, and the existence of algorithmics as constituent domain of computer science alongside others, like the theory of languages or the theory of robots.
Our research studies the question of the introduction of elements of algorithmics and programming in the secondary mathematical teaching. It relies on epistemological and institutional analyses that show, on one hand that the notions of loop and computer variable are built at the same time as the architecture of the machine is transformed. On the other hand, it testifies the difficult presence of the elements of algorithmics and programming in secondary teaching in France and in Vietnam. The results of these analyses build the conception and the realization of a didactic engineering in a computer environment. It is conceived as an experimental genesis of the machine of Von Neumann and programming through the writing of the successive messages (programs) to machines endowed with different characteristics. This conception uses the tools of the theory of Didactic Situations to organize, from a fundamental situation of algorithmics and programming, a first encounter with different types of memories of the machine, in particular erasable memories. This first encounter allows for the emergence of the notion of computer variables and the loop in our didactic engineering. For the need of our research, we constructed an emulated calculator, named Alpro. This emulator is based on the model of calculator existing in the secondary teaching of the two countries and having the additional capacity to record the history of the pressed keys at the time of a calculation.

Keys words

Epistemological and institutional analyses, Theory of the Didactic Situations and didactic engineering, instrumental genesis, erasable memory, programs recorded, architecture of a machine, problem of tabulation of a numerical function, algorithm, programming of an algorithm, computer variable, loop.

THÈSE EN CO-TUTELLE

Présentée par

NGUYỄN Chí Thành

Pour obtenir les titres de

Docteur de l'Université Joseph Fourier, Grenoble et Docteur du Viêt-nam

Spécialité : DIDACTIQUE DES MATHEMATIQUES

ÉTUDE DIDACTIQUE DE L'INTRODUCTION D'ÉLÉMENTS D'ALGORITHMIQUE ET DE PROGRAMMATION DANS L'ENSEIGNEMENT MATHÉMATIQUE SECONDAIRE À L'AIDE DE LA CALCULATRICE

Soutenue le 16 Décembre 2005

Composition du jury :

Annie **BESSOT**	Co-directrice de thèse
Philippe **JORRAND**	Examinateur
Jean-Baptiste **LAGRANGE**	Rapporteur
LÊ THỊ Hoài Châu	Examinatrice
LÊ VĂN Tiến	Rapporteur
NGUYỄN BÁ Kim	Co-directeur de thèse
Alain **BIREBENT**	Invité

Thèse préparée au sein des équipes

Didactique des Mathématiques (DDM), Laboratoire Leibniz – IMAG

Formation de Troisième cycle, École normale supérieure de Hanoï

A la mémoire de mon père

Remerciements

Annie Bessot a accepté de diriger cette thèse après avoir encadré mon mémoire en Master 2. Rigoureuse et efficace, avec son expérience et son dévouement elle m'a toujours encouragé, guidé et soutenu. L'encadrement d'un thésard est un travail difficile et celui d'un thésard étranger l'est beaucoup plus. Cette thèse n'aurait pas pu être menée à son terme sans ses conseils, son aide et ses remarques pertinentes en particulier durant la rédaction du manuscrit. Je l'en remercie profondément et affectueusement ici.

Mes remerciements vont également à Alain Birebent qui a grandement participé à l'avancée de mon travail. Les réunions à trois, avec lui et avec Annie, ont toujours été pour moi des moments scientifiquement fructueux et amicalement enrichissants.

Je voudrais aussi remercier Monsieur Philippe Jorrand qui a suivi ce travail depuis mon arrivée en Master 2. Les discussions avec lui ont été très utiles en ce qui concerne l'informatique et m'ont permis de progresser en ce domaine.

Messieurs Jean-Baptiste Lagrange et Lê Văn Tiến se sont rendus disponibles en acceptant d'être rapporteurs de cette thèse. Leurs commentaires pertinents m'encouragent dans la voie de la recherche en Didactique Des Mathématiques. Qu'ils veuillent bien recevoir mes respectueux remerciements.

Je voudrais remercier Monsieur Nguyễn Bá Kim pour avoir accepté d'être codirecteur de ce travail. Il a ainsi facilité mon intégration dans le cadre d'une thèse en cotutelle entre le Viêt-nam et la France.

Je suis très honoré par la présence de Madame Lê Thị Hoài Châu qui a accepté d'être présidente de mon jury.

Je voudrais remercier Christophe Daudin, Olivier Tawin, Nguyễn thị Thủy, Lê Thành Thái ainsi que leurs élèves pour m'avoir ouvert leur classe. Christophe et Thái ont, en particulier, assumé le rôle d'enseignant pour une partie de l'expérimentation de ma thèse.

Mes vifs remerciements vont aussi à l'Agence Universitaire de la Francophonie et à la région Rhône-Alpes dont les bourses m'ont permis d'obtenir des ressources financières pour effectuer cette étude.

Que Monsieur Đinh Quang Thú, directeur du service des relations internationales à l'Ecole Normale Supérieure de Hanoi (ENSH) trouve ici ma reconnaissance pour toutes ses interventions sur le plan administratif en faveur de la mise en place d'une convention de cotutelle entre ENSH et l'Université de Joseph Fourier de Grenoble.

Je n'oublie pas le personnel du Laboratoire Leibniz et celui de l'IREM de Grenoble, particulièrement mes collègues de l'équipe DDM dont l'ambiance chaleureuse a toujours contribué à m'encourager durant les moments difficiles. Je pense ainsi très fort à Claude, Madeleine, Jean-Luc, Afonso, Khanh, Laetitia et Trung.

Je remercie tous les membres de ma famille et aussi de ma « belle famille » qui m'ont toujours accompagné pendant ces années de préparation de ma thèse.

Une pensée spéciale à Diệu Hương, ma chère femme qui a vécu aussi intensément que moi ces trois années. Merci de m'avoir toujours fait confiance et soutenu.

Table des matières

Introduction

L'introduction d'éléments d'informatique dans l'Enseignement Mathématique Secondaire (EMS) connaît, depuis une vingtaine d'années, des tentatives diverses dans de nombreux pays, dont la France et le Viêt-nam qui sont les deux pays concernés par cette étude.

Les arguments généralement avancés par les promoteurs d'une telle introduction sont de deux ordres :

- le caractère fondamental et universel de tels éléments, car ils sont à l'articulation de plusieurs disciplines scientifiques dont la logique, les mathématiques et l'informatique ;
- leur apport à l'enseignement des mathématiques à la fois dans l'appréhension des technologies informatiques et dans la transformation de certains concepts mathématiques.

Mais par delà l'accord sur ces arguments, on repère deux tendances (au moins dans les deux pays concernés), pour introduire des éléments d'informatique dans l'enseignement secondaire :

- au sein de l'informatique en tant que discipline scientifique enseignée ;
- ou dépendant de l'organisation mathématique dans EMS.

Récemment en France, la commission de réflexion sur l'enseignement des mathématiques, dite commission Kahane, a proposé « *d'introduire une part d'informatique dans l'enseignement des sciences mathématiques et dans la formation des maîtres* » en faisant évoluer progressivement les contenus « *pour intégrer de nouveaux objets et notions d'algorithmique et programmation* ». (Rapport de la commission Kahane 2001)[1]

En affirmant la nécessité de répandre l'usage d'outils informatiques dans l'enseignement des mathématiques, cette proposition s'inscrit en continuité par rapport aux choix noosphériens passés et actuels. Mais, en rupture avec ces mêmes choix, elle vise à accentuer la construction de liens théoriques qui unissent l'informatique aux mathématiques.
La comparaison avec le rapport d'une commission précédente permet d'apprécier la nature de cette rupture :

> Nous ne pensons pas que l'informatique doive être enseignée comme discipline (théorique) en tant que telle à ces niveaux de formation. L'informatique enseignée à ce niveau contient des risques de formalisme encore plus graves que les mathématiques. L'argument des élèves mauvais en mathématiques qui se rattraperaient par l'informatique est peu fondé. Par contre l'introduction d'outils informatiques peut « débloquer » des élèves en difficulté et en motiver d'autres. (Rapport de la mission Dacunha-Castelle, 1989)

La commission Kahane fait d'ailleurs le procès de cette introduction d'outils informatiques sans ambition d'enseignement sur les objets de l'informatique :

> Les programmes de mathématiques des années 90 stipulent que l'élève devra maîtriser l'usage d'une calculatrice scientifique (seconde et première) et d'une calculatrice programmable (terminale). Si cette requête est affichée dans l'introduction du document, il n'en est plus fait mention dans l'évocation des contenus et des activités. Comment le législateur imaginait-il cet apprentissage ? Il est peut-être temps de

[1] Ce rapport est entièrement disponible sur le site de la SMF (Société de Mathématique de France) http://smf.emath.fr/Enseignement/CommissionKahane/

faire entrer ce souhait dans les faits en s'en donnant les moyens. (Rapport de la commission Kahane, 2001)

Dans le même temps au Viêt-nam, Nguyen Van Trang et Ta Duy Phuong (2000), mathématiciens et noosphériens, déclarent à un colloque sur l'enseignement des mathématiques, organisé par le ministère de l'éducation et de la formation (MEF) :

La chose la plus importante à enseigner n'est pas enseigner des calculs avec des opérations arithmétiques ou enseigner à utiliser les programmes déjà intégrés dans la calculatrice (statistiques, résolution des équations etc.) mais enseigner *la pensée algorithmique*. La pensée algorithmique, très importante dans cette ère de la technologie informatique, se manifeste à travers plusieurs types d'exercices dont les *contenus sont ancrés profondément en mathématiques*. […] La pensée algorithmique, grâce à la calculatrice, sera un lien entre deux disciplines étroitement liées l'une à l'autre, mais qui sont enseignées d'une manière séparée à l'école. Ces deux disciplines sont mathématiques et informatiques. […] Plusieurs algorithmes (recherche des nombres premiers grands, calcul des formules de récurrence qui étaient impossibles de réaliser dans la pratique sont à notre portée grâce à la calculatrice. […] La calculatrice est un outil pertinent aidant la construction des notions mathématiques (limite, intégrale). Une nouvelle méthode s'ouvre dans la construction et dans la réalisation des fiches pédagogiques qui est la théorie en association avec la pratique » (p 30, traduit par nous).

Mais le discours sur la pensée algorithmique a un caractère général qui ne désigne pas les objets de savoir à enseigner.
Au contraire, la commission Kahane, dans son rapport, désigne à l'enseignement « *dans le cadre des sciences mathématiques* » ce qu'elle appelle les concepts de base d'algorithmique et de programmation :

- À propos de programmation : des structures de contrôle (boucles et branchements) et la récursivité ;
- À propos d'algorithmique : des structures de données et la complexité des algorithmes.

Regardons de plus près ces objets désignés comme concepts de base d'algorithmique et de programmation pour éclairer d'une part, ce qui unit et ce qui sépare l'algorithmique de la programmation, d'autre part les choix possibles d'objets de base en vue de leur introduction dans EMS.

I. Quelques généralités sur les concepts de base d'algorithmique et de programmation

Un programme est un texte, l'histoire qu'il raconte un algorithme, la programmation son écriture. (Ganascia 1998)

I.1. Algorithmique

Dans l'Encyclopaedia Universalis, Hebeinstrait donne la définition suivante du mot algorithme :

Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données, pour arriver en un nombre fini d'étapes, à certain résultat, et cela indépendamment des données. (Hebeinstrait, Encyclopaedia Universalis, tome 12, p. 306)

L'algorithmique est la science des algorithmes :

L'objet de l'algorithmique est la conception, l'évaluation et l'optimisation des méthodes de calcul en mathématiques et en informatique. (Flajoret, Encyclopaedia Universalis, tome 1, p. 814)

Les objets principaux en algorithmique sont :

- Des algorithmes fondamentaux : algorithmes mathématiques, algorithmes de tri, algorithmes de recherche, algorithmes géométriques, algorithmes de graphes, etc. ;
- Des structures de données : tableaux, arbres, ensembles, listes, fichiers, etc. ;

- L’analyse de la complexité des algorithmes, classes de problèmes NP, etc. ;
- Des techniques de conception d’algorithmes : programmation dynamique, algorithmes gloutons, etc.

Ainsi l’algorithmique cherche à étudier l’algorithme indépendamment de sa mise en œuvre dans une machine.

> Sans entrer dans les détails mathématiques, on peut dire que lorsque l'on calcule l'efficacité d'un algorithme (*sa complexité algorithmique*), on cherche davantage à connaître l'évolution du nombre d'instructions de base en fonction de la quantité de données à traiter (par exemple, dans un algorithme de tri, le nombre de lignes à trier), que le coût exact en secondes et en quantité de mémoire. Baser le calcul de la complexité d'un algorithme sur le temps qu'un ordinateur particulier prend pour effectuer ledit algorithme ne permet pas de prendre en compte la structure interne de l'algorithme ni la particularité de l'ordinateur : selon sa charge de travail, la vitesse de son processeur, la vitesse d'accès aux données ou même l'exécution de l'algorithme (qui peut faire intervenir le hasard) le temps d'exécution ne sera pas le même. (Wikipedia 2005)

Dans le travail algorithmique, il est nécessaire d’organiser, de classer et de désigner les données du problème sur lequel on travaille et dont l’algorithme est une solution. Parmi les structures de données présentées comme élémentaires dans les cours universitaires introductifs à l’informatique, on trouve listes, tableaux, piles, files, fichiers dans lesquels intervient une notion de base, celle de *variable*.

I.2. Programmation

La programmation dans le domaine informatique est l'ensemble des activités qui permettent l'écriture des programmes informatiques pour leur exécution dans une machine.

> Programme informatique : la liste des instructions auxquelles la machine devra obéir, dans l’ordre de leur exécution […]. On le charge dans la mémoire de la machine, où elle puisera les instructions au fur et à mesure de leur exécution, à sa propre vitesse. (Arsac, Encyclopaedia Universalis, tome 19, p. 31)

Les informaticiens utilisent fréquemment le mot d’origine anglaise « implémentation » pour désigner cette mise en œuvre. L'écriture en langage informatique est aussi fréquemment désignée par le terme de « codage », qui n'a ici aucun rapport avec la cryptographie, mais qui se réfère au terme « code source » pour désigner le texte, en langage de programmation, constituant le programme. L'algorithme devra être plus ou moins détaillé selon le niveau d'abstraction du langage utilisé.

Les langages de programmation permettent de définir les ensembles d'instructions effectuées par l'ordinateur lors de l'exécution d'un programme. Il existe des milliers de langages de programmation, la plupart d'entre eux étant réservés à des domaines spécialisés. Ils font l'objet de recherches constantes dans les universités et dans l'industrie.

Un programme informatique indique à un ordinateur ce qu'il doit faire. Il s'agit d'un ensemble d'instructions qui doivent être exécutés dans un certain ordre par un processus.

> Un ordinateur sans programme ne fait absolument rien. En fait, c'est même la possibilité de suivre un programme enregistré qui sert, d'un point de vue historique, à distinguer un ordinateur d'une simple machine à calculer. Le premier ordinateur est donc le Manchester Mark I, premier calculateur à programme enregistré. (Arsac, op.cité)

Une structure de contrôle permet de diriger l’exécution d’un algorithme par une machine. On distingue, dans les cours universitaires introductifs à l’informatique, trois structures de contrôle dans une programmation non structurée par la numérotation des instructions :

- séquentialité : les instructions sont exécutées les unes après les autres ;
- les branchements exprimés par une instruction conditionnelle ;
- l’itération qui consiste à répéter un groupe d’instructions, appelé boucle.

II. L'algorithmique et la programmation dans EMS actuelle

De façon très synthétique, dans EMS actuel, on constate qu'il y a :
- des algorithmes mais pas d'algorithmique enseignée (en France et au Viêt-nam)
- des programmes de calcul pour des calculatrices programmables ou des tableurs mais pas de programmation enseignée (en France) ;
- des machines-ordinateurs mais l'architecture d'une telle machine n'est pas un objet d'enseignement (en France).

Un tel constat ne renvoie pas essentiellement à des faiblesses du législateur comme le suggère le rapport de la commission Kahane mais aux conditions écologiques et économiques du système didactique concerné. C'est ce que nous rappelle Chevallard (1986) :

> On réforme les programmes, on y introduit de nouveaux "objets d'enseignement". Or, bien souvent, ceux-ci se révèlent trop "gros". Et parce qu'il s'aperçoit que leur gestion dans la classe est lourde, invalidante, impossible, l'enseignant doit bien vite les apprêter, les "dégraisser", les calibrer, voire se résoudre à les écarter. Le problème didactique ainsi posé est, d'une certaine manière, bien connu. Mais, pour l'énoncer correctement, il convient de voir qu'il ne surgit pas seulement de manière anecdotique (aussi ne parlerai-je pas, ici, de la trop fameuse "droite affine en quatrième"), mais bien de manière systématique. Il convient de ne pas y voir seulement l'effet de quelque décision irréfléchie des rédacteurs des programmes, mais bien de la conséquence régulière des lois spécifiques du fonctionnement didactique.

Ce constat nous conduit à un premier questionnement :

Quelles sont les conditions qui peuvent permettre à des objets nouveaux comme les structures de données et les structures de contrôle, de vivre dans EMS ? Mais quels objets élémentaires[1] *nouveaux ? Avec quels problèmes mathématiques ? Avec quelles organisations mathématiques et didactiques ?*

Nous allons développer ces questions en fixant :
- un domaine des mathématiques : le Calcul Numérique
- une technologie informatique : la calculatrice

Justifions nos choix.

III. Calcul numérique, algorithmes et instruments de calcul

III.1. Calcul numérique et algorithmes

Le calcul numérique est un lieu privilégié de production d'algorithmes notamment d'algorithmes itératifs où il y a répétition d'un invariant de calcul (division de deux nombres entiers, approximation d'une racine d'une équation numérique, etc.).
La répétition s'appuie sur une discrétisation des ensembles de nombres et sur la détermination d'actions effectuables et reproductibles. Sa formulation, en vue d'obtenir un résultat numérique ou une preuve, produit des algorithmes de calcul.

Dans les praxéologies de calcul numérique élaborées par EMS (l'approximation décimale, la construction d'une table numérique d'une fonction numérique etc.) où sont présents des algorithmes, la répétition peut être outillée par divers objets mathématiques comme les formules et les fonctions, les suites et la notion de récurrence.

[1] élémentaire non pas au sens de « simple » mais au sens d'élément organisateur d'un savoir.

III.2. Calcul numérique et instruments de calculs

Une part plus ou moins importante de la répétition peut être déléguée à un ou des instruments de calculs qui permettent à un opérateur humain de réaliser un calcul par l'intermédiaire d'un algorithme explicite ou cristallisé dans l'instrument, comme une table numérique, un boulier, une calculatrice non programmable, etc. Cette possibilité de déléguer le calcul répétitif a été exploitée dans chaque tentative d'introduire un instrument de calcul nouveau dans l'enseignement des mathématiques pour renforcer l'effectivité du calcul numérique.

L'introduction de l'instrument calculatrice dans une institution d'enseignement des mathématiques comme EMS procède d'une genèse institutionnelle qui réorganise les savoirs et les techniques de calcul numérique. Inversement on peut s'attendre à ce que l'introduction d'éléments d'algorithmique et de programmation modifie l'instrumentation des calculs numériques et la prise en charge institutionnelle de leur effectivité.

III.3. Choix des notions de boucle et variable

L'importance des algorithmes itératifs dans EMS nous conduit à nous centrer sur la notion de boucle comme candidate à être une structure de contrôle organisatrice des autres structures de contrôle et donc sur la notion de variable informatique qui lui est intrinsèquement liée du point de vue de la programmation[1]. Les raisons de ces choix seront approfondies au cours de notre recherche.
Nous complétons notre questionnement initial par les questions supplémentaires :

Comment l'introduction de ces éléments d'algorithmique et de programmation peut-elle s'appuyer sur l'intégration actuelle de la calculatrice dans l'enseignement secondaire des mathématiques ? Comment peut-elle transformer sa nature d'instrument ? Comment peut-elle s'appuyer sur les praxéologies de calcul numérique ? Comment peut-elle les transformer ?

IV. Les choix théoriques et méthodologiques

IV.1. L'analyse de la calculatrice comme instrument

La notion d'instrument et celle d'activité mathématique instrumentée sont au cœur de cette recherche. Depuis une quinzaine d'années ces notions font l'objet de nombreuses recherches en Didactique des mathématiques, notamment sur le thème de l'intégration d'objets techniques complexes. Ces recherches se développent principalement selon deux points de vue :

- un point de vue psychologique qui analyse et conceptualise l'activité instrumentée du point de vue du fonctionnement cognitif de l'individu ;
- un point de vue anthropologique qui analyse l'activité instrumentée en termes de pratiques et de théories portées, valorisées et normées par une institution, et que reproduit l'individu en tant que sujet de l'institution.

Bien qu'en empruntant à des cadres théoriques globaux bien distincts, ces deux axes offrent des ressources que notre projet peut exploiter et conjuguer.

Le point de vue psychologique

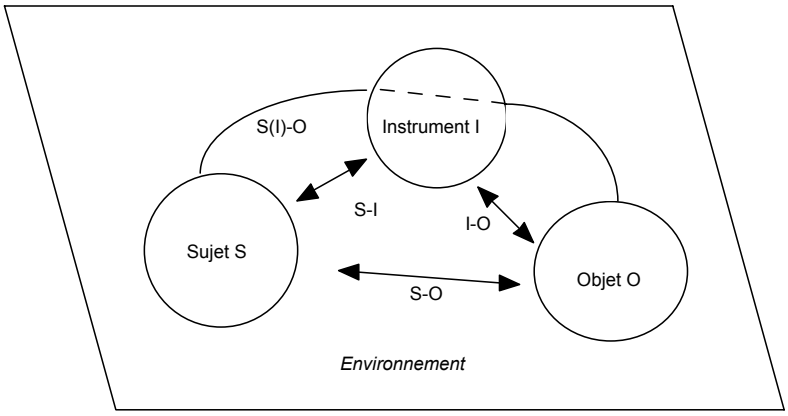
Celles qui intéressent le plus une recherche relative à l'enseignement proviennent des travaux en ergonomie cognitive (cf. Rabardel 1995 et Vérillon 1996).

[1] Dans le style impératif.

Voici, en quatre points, une présentation de ce point de vue, extraite d'Artigue et Lagrange (1998) :

> 1. Un objet technique n'est pas d'emblée un instrument, même si l'on tend à le considérer comme tel. C'est d'abord un objet, un artefact, selon la terminologie utilisée par Rabardel (1996), pour rester le plus neutre possible. C'est dans l'évolution de nos rapports à l'objet que se construit l'instrument, au cours d'un processus de genèse instrumentale, en général complexe.
> 2. Cette genèse instrumentale est à la fois dirigée vers l'objet et le sujet, dans un double processus d'*instrumentalisation* et d'*instrumentation*. L'instrumentalisation, dirigée vers l'objet, conduit à le personnaliser, à le transformer éventuellement, à lui conférer des fonctionnalités dont certaines pouvaient être *a priori* non prévues par le concepteur. […]. L'instrumentation, dirigée vers le sujet, conduit à élaborer de façon autonome ou à s'approprier socialement des schèmes d'action instrumentée […].
> 3. Les schèmes sont plurifonctionnels. Mis en jeu dans des situations précises, ils aident à comprendre (c'est là leur fonction *épistémique*), ils aident à agir, à transformer, à résoudre (c'est là leur fonction *pragmatique*), ils aident enfin à organiser et contrôler l'action (c'est là leur fonction *heuristique*).
> 4. L'activité instrumentée influe à la fois sur les modes d'accès aux connaissances et sur les connaissances élaborées elles-mêmes :
> - d'une part, via les contraintes introduites par l'artefact, et il semble utile à ce niveau de distinguer entre les contraintes liées à la structure de l'artefact lui-même (les contraintes internes), celles liées aux actions et transformations qu'il permet (les contraintes de commande), celles enfin liées à la façon dont il tend à organiser l'activité (les contraintes d'organisation),
> - d'autre part, via les potentialités nouvelles offertes à l'action du sujet par l'artefact.
> Il s'ensuit un rapport nécessairement dialectique entre instrumentation et apprentissage des mathématiques. (Actes du Colloque francophone européen, 1998, pages 18 et 19)

Nous retenons particulièrement la double notion d'instrumentalisation et d'instrumentation. Rabardel (1996) et Vérillon (1996) la construisent au sein d'une modélisation des **s**ituations d'**a**ctivité **i**nstrumentée. Le modèle qu'ils appellent SAI est présenté par cette figure :



Vérillon (1996) explique alors :

> L'instrumentation concerne donc l'élaboration des rapports S-I : le sujet doit construire les schèmes, les procédures, les opérations nécessaires à la mise en œuvre de l'artefact. Il peut, par exemple, importer dans la situation des rapports S-I construits dans d'autres contextes avec d'autres artefacts ou, au contraire, construire ces nouveaux rapports de manière exploratoire, ou encore, les élaborer par imitation.
> L'instrumentalisation concerne, quant à elle, la construction des rapports I-O. Le sujet attribue à l'instrument une possibilité d'agir sur O et construit les propriétés fonctionnelles qui permettent l'actualisation de cette possibilité d'action. (Ibid. page 5)

Les algorithmes itératifs résident au cœur du développement de nouveaux rapports aux objets mathématiques ou informatiques que nous cherchons à analyser.

Supposons que l'objet O dont parle Vérillon soit l'objet « variable » dans un calcul d'une expression numérique, le sujet S un élève de EMS et l'instrument I une calculatrice ou une table numérique. La dialectique instrumentation / instrumentalisation décrit la façon dont la

présence de l'instrument influe sur la constitution d'un rapport S-O de l'élève au calcul en le « doublant » (ou en l'enrichissant) d'un rapport S(I)-O qui apparaîtra, voire s'imposera, dans toutes les situations où la calculatrice sera disponible. C'est en ce sens que l'on doit affirmer qu'un instrument n'est pas neutre à l'égard de l'organisation et de l'exécution d'un calcul numérique, ni épistémologiquement ni didactiquement. Il ne s'intègre pas naturellement dans l'activité calculatoire d'un individu. L'instrument, dit Prudhomme (1999) :

> […] est une construction de son activité dans des situations données et non quelque chose de disponible qu'il suffit de « fournir » pour que l'individu l'associe à son action. (page 29)

Le point de vue anthropologique

Comme le dit Birebent (2001) :

> La genèse instrumentale que décrit Vérillon apparaît donc, pour chaque individu, comme un construit à la fois individuel et social, évoluant au gré de ses rencontres avec les situations qui lui posent problème. Cela laisse imaginer des caractères et des niveaux très divers de genèse instrumentale pour une même classe de problèmes. Mais dans une institution didactique de EMS, la rencontre d'un élève avec un problème mathématique ne relève pas d'un processus chaotique ; elle est en quelque sorte programmée.

C'est ce que nous rappelle Chevallard (1992) :

> Dans la plupart des pratiques sociales, l'objet est "jugé" au résultat de l'action qu'il permet, et aux coûts qu'elle suppose. Dans les pratiques de transmission de savoir, l'action qu'il autorise fait partie de *la manipulation du savoir enseigné*, et s'intègre dans le *rapport au savoir* des acteurs de la relation didactique, enseignant et enseignés. Elle doit donc apparaître comme *légitime*, et entre à ce titre dans le registre des enjeux didactiques. Il n'y a ainsi ni mystère ni scandale à voir la calculette, par exemple, prospérer entre les mains de l'épicier, de la ménagère, et de l'élève, mais hors de la classe. Et de la voir, dans le même temps, susciter l'interrogation dubitative de maints enseignants de mathématiques (pas de géographie, ou de physique, bien sûr, car elle trouve, en ces disciplines, un statut épistémologique et didactique assuré, celui d'aide au calcul). Dans un cas, dans la plupart des cas, son usage va sans problèmes (autres que matériels : prix, fiabilité, maniabilité, etc.). Dans l'autre cas, son emploi est exactement problématique et les scénarios de son intégration à la classe de mathématiques sont l'objet d'études, d'expérimentations, de colloques. A-t-on jamais vu épiciers tenir congrès sur ce thème ?

Ainsi quand Rabardel (1995) écrit :

> L'artefact est institué comme instrument par le sujet qui lui donne le statut de moyen pour atteindre le but de son action.

Nous pensons, comme Birebent (2001) que cette institutionnalisation chez l'individu est en fait un produit de l'institution de laquelle l'individu est sujet et qui définit l'action. C'est dans l'institution EMS que l'élève découvre certaines tâches calculatoires et, en même temps que ces tâches, une ou des techniques, un ou des énoncés théoriques.

IV.2. Analyse institutionnelle comparative

Nous allons croiser dans cette recherche des analyses de deux systèmes d'enseignement secondaires et de deux systèmes d'enseignement supérieur : EMS et l'enseignement informatique supérieur, en France et au Viêt-nam.

Nous allons le faire selon deux axes, diachronique et synchronique, l'analyse diachronique permettant de comprendre l'état actuel d'un système à partir de son évolution.

La comparaison synchronique quant à elle a pour but :

- De dévoiler et de caractériser les produits différents et semblables des transpositions didactiques de mêmes objets de savoir ;
- De questionner ces similitudes et ces différences en termes de conditions et de contraintes ;

- D'initier un répertoire de praxéologies existantes et d'envisager leurs développements possibles

> Face à quelques obligations d'agir, en effet nous commençons par observer et analyser […] la manière de faire de quelque autrui […]. Puis nous évaluons ce qu'observation et analyse auront ainsi révélé […], avant de développer notre propre « solution » en essayant d'améliorer, sur certains points jugés négativement, la « solution » observée. (Chevallard 1998)

IV.3. Ingénierie didactique

Dans cette recherche, suivant en cela une méthodologie propre à la didactique des mathématiques, nous avons conçu et expérimenté une ingénierie didactique. L'ingénierie didactique est à la fois une réalisation didactique en classe et un outil de recherche construit dialectiquement avec des analyses épistémologiques et institutionnelles préalables. Elle organise une genèse expérimentale à partir d'un jeu sur les variables didactiques d'une situation fondamentale des savoirs informatiques choisis (Brousseau 1998) : variable et boucle.

> Un champ de problèmes peut être engendré à partir d'une situation par la modification des valeurs de certaines variables qui, à leur tour, font changer les caractéristiques des stratégies de solution (coût, validité, complexité…etc.) […]
> Seules les modifications qui affectent la hiérarchie des stratégies sont à considérer (variables pertinentes) et parmi les variables pertinentes, celles que peut manipuler un professeur sont particulièrement intéressantes : ce sont les *variables didactiques*. (Brousseau 1982)

Elle a été expérimentée dans EMS des deux pays. Son élaboration a commencé avec notre mémoire de DEA (Nguyen 2002) et a connu une longue maturation ponctuée de pré-expérimentations tant au Viêt-nam qu'en France.

Cette ingénierie didactique nous a permis d'agir rationnellement sur chacun des systèmes d'enseignement (raison basée sur des connaissances didactiques préétablies) et de mettre à l'épreuve de la réalité des hypothèses de recherche.

L'expérimentation de l'ingénierie nous a conduit à rencontrer les objets complexes de la réalité (de l'enseignement) et à essayer de prendre en compte dans sa réalisation les comportements des enseignants. En l'absence d'un cadre théorique spécifique, cela s'est fait par des négociations relativement informelles, avant et après les séances (ces dernières ont été enregistrées).

Nous avons rencontré, certainement à cause du thème choisi pour l'ingénierie didactique, « Introduction dans EMS d'éléments d'algorithmique et de programmation », une situation paradoxale :

- Une forte incitation des institutions ;
- Et des craintes exprimées par les enseignants.

En effet, si les enseignants de Lycée contactés ont accepté de libérer du temps d'enseignement pour la réalisation de l'ingénierie, certains ont refusé d'endosser la responsabilité d'enseignant dans l'expérimentation.

Cette ingénierie est, pour nous une ouverture sur le problème d'une formation des enseignants à la hauteur des incitations des institutions.

Notre recherche s'organise selon le plan suivant.

V. Plan de recherche

Partie A. Algorithmique et programmation : le point sur les recherches en didactique et analyse institutionnelle

Chapitre A1. Travaux de références en Didactique des Mathématiques ;

Chapitre A2. Présence des notions d'algorithme, de boucle et de variable dans EMS en France et au Viêt-nam ;

Chapitre A3. La tentative d'introduction de l'informatique dans EMS au Viêt-nam.

Conclusion de la partie A

Partie B. Algorithmique et programmation : une enquête épistémologique

Chapitre B1. Emergence d'un enseignement d'algorithmique et de programmation ;

Chapitre B2. Machines et programmes ;

Chapitre B3. Du programme de Ada au langage évolué ;

Conclusion de la partie B

Partie C. Algorithmique et programmation : choix macro-didactiques pour une ingénierie didactique

Chapitre C1. Une situation fondamentale de l'algorithmique et de la programmation : la tabulation d'une fonction numérique par une machine

Chapitre C2. Quelle machine dans EMS ? Vers une conception d'une calculatrice générique Alpro

Conclusion de la partie C

Partie D. Conception, réalisation et analyse de l'ingénierie didactique

Chapitre D1. Analyse de la situation 1 : disponibilité et différenciation des mémoires variables A, B, C et Ans

Chapitre D2. Analyse de la situation 2 : programme à une machine (Alpro ; CALCULATOR) proche de la machine analytique de Babbage

Chapitre D3. Analyse de la situation 3 : Co-évolution du langage et de machine (Alpro ; CALCULATOR II), machine ordinateur de Von Neumann

Chapitre D4. Deux études de cas à travers l'ingénierie didactique

Conclusions générales et perspectives

Partie A

Algorithmique et programmation
Le point sur les recherches en didactique
Analyse institutionnelle

Chapitre A1
Travaux de références en Didactique des Mathématiques

Introduction

Nous mettons en lumière ici certains résultats de travaux de recherches qui ont porté directement ou indirectement sur l'acquisition de la notion d'algorithme et de premières notions d'informatique dans l'enseignement secondaire. Les recherches auxquelles nous nous référons ont été menées ces deux dernières décennies en France dans les cadres théoriques de la didactique des mathématiques ou de la psychologie cognitive.

Ce chapitre comporte deux parties :

- la première concerne des recherches qui interrogent la présence d'algorithmes et d'éléments de programmation dans l'enseignement des mathématiques en France et au Viêt Nam ;
- la deuxième concerne des recherches qui se présentent comme des tentatives d'introduction d'éléments de programmation dans une option informatique en France dans les années 1980.

I. Présence d'algorithmes et de la programmation dans l'enseignement des mathématiques

Dans son travail, Rajoson (1988) étudie les conditions de la vie de certains algorithmes. L'exemple de la disparition de l'algorithme de Héron dans EMS lui permet d'avancer que la vie des algorithmes est rendue difficile dans l'enseignement des mathématiques s'ils sont des isolats au sein des mathématiques enseignées :

> Pour être viable au sein d'un corpus de savoir (savant ou enseigné), un élément de savoir donné doit pouvoir y apparaître […] comme partie d'un tout structuré. (Rajoson 1988, p. 135)

À la suite de Rajoson, d'autres études se sont penchées sur la vie d'algorithmes dans deux habitats principaux au Lycée en France : celui de l'Analyse et celui de l'Arithmétique.

La contre-réforme en France est marquée par la volonté noosphérienne d'introduire dans EMS une problématique de l'approximation numérique au sein de l'Analyse. Ceci conduit Birebent (2001) et Lê Van (2001) à étudier au travers des différents programmes la vie d'algorithmes de calcul approché dans EMS (avant 2000) et de leur programmation. Lê Van (2001) réalise son étude non seulement en France mais aussi au Viêt-nam.

Nguyen et Bessot (2003) en se centrant sur la programmation questionnent dans le domaine de l'analyse « la prise en compte des notions de boucle et de variable informatique dans l'enseignement des mathématiques au lycée ».

La réintroduction de l'arithmétique en Terminale S conduit Ravel (2003) à interroger la vie dans ce domaine des algorithmes de calcul numérique.

Nous nous intéresserons ici aux algorithmes susceptibles d'être associés à un programme avec boucles. Les algorithmes non itératifs comme le calcul du discriminant dans la résolution d'une équation du second degré resteront hors du champ de cette revue de travaux.

I.1. La vie des algorithmes et de la programmation dans EMS en France dans l'habitat de l'Analyse

I.1.1. Lê Van Tien (2001)

Dans son travail de thèse, Lê Van repère d'abord les réformes entreprises en France et au Viêt-nam qui ont marqué l'évolution des deux systèmes éducatifs. Puis il analyse les discours et les écrits noosphériens précédant ces réformes, ainsi que les programmes et les manuels des mathématiques dans les deux pays depuis les années 70 jusqu'aux années 90. Certains des résultats de ces analyses portent sur l'objet « la programmation » dans EMS.

A propos de la contre-réforme des années 80 en France, il montre que dans les intentions des noosphériens la calculatrice conditionne l'existence de moments expérimentaux (voulus par les réformateurs) au sein de l'enseignement des mathématiques, ces moments expérimentaux étant tout particulièrement associés au numérique. De ce fait, l'étude d'algorithmes de calcul devient, *en intention*, l'un des objectifs de la contre-réforme :

> L'objet « calculatrice » apparaît comme une condition écologique fondamentale à la mise en œuvre effective dans l'enseignement du moment expérimental, en particulier de l'étude par l'élève des exemples numériques. […] Autrement dit, l'obligation sociale constitue le second élément qui donne la légitimité à l'introduction des calculatrices dans les classes. […] L'étude d'algorithme devient ainsi l'un des objectifs importants de l'enseignement des mathématiques. (Le Van 2001, p. 146)

Il parle d'une entrée timide du domaine algorithmique / programmation / informatique dans EMS :

> L'approche de l'Informatique passe ainsi essentiellement par l'utilisation des calculatrices programmables et est justifiée par l'insistance des programmes sur l'aspect algorithmique des méthodes de l'Analyse. Les éléments du domaine algorithmique/programmation/informatique font ainsi une entrée timide au sein du cursus mathématique à enseigner (Le Van 2001, p. 180)

Il y a une forte relation entre algorithmes et calculatrices dans EMS durant toute la contre-réforme en France :

> Dans l'enseignement des mathématiques, les calculatrices sont incontournables, d'une part comme outil permettant la mise en œuvre effective du moment expérimental et la mise en œuvre de l'aspect algorithmique des mathématiques, d'autre part comme réponse aux besoins sociaux de l'informatique. (Le Van 2001, p. 201)

Lê Van montre que l'évolution des intentions de la contre-réforme jusqu'à la fin des années 90, maintient la présence de l'aspect algorithmique et d'éléments de programmation dans les programmes sans rendre exigibles aux élèves des connaissances déterminées sur les algorithmes et les programmes. Il est alors prévisible que leur place soit réduite dans les pratiques réelles :

> Au lycée, les commentaires des programmes de toutes les classes réservent un paragraphe spécifique pour parler de l'algorithmique. La mise en valeur des aspects algorithmiques des problèmes étudiés continue à être présente, mais aucune connaissance spécifique sur cette question n'est exigible des élèves. […] La capacité de programmation, qui n'était exigée, à l'époque précédente, qu'à la fin de Première, est mentionnée désormais dès la classe de Seconde et dans les classes suivantes. (Le Van 2001, p. 188)

En effet, l'étude de l'évolution des méthodes de calcul approché des racines d'une équation au cours des différentes réformes, permet à Lê Van d'attester la réduction de la place des algorithmes dans les manuels, mais aussi de celle de la programmation qui se ramène à la présence de programmes préfabriqués :

> Les manuels de Première Belin, Hatier, Fractale et Transmath (édition 1995) qui se limitent aux méthodes de Dichotomie ou de Balayage, ne laissent subsister de l'algorithme que le calcul des premiers termes et la donnée de programmes tout faits à implémenter sur une calculatrice programmable ou un ordinateur1 (Basic ou Turbo Pascal). Mêmes les noms « dichotomie » et « balayage » ont disparu du manuel de Première Transmath. (p. 188)

I.1.2. Alain Birebent (2001)

Dans son travail de thèse, Birebent (2001) conduit une étude écologique de plusieurs types de tâche qu'il qualifie d'emblématiques de l'Analyse réelle en tant qu'entrée dans le champ de l'approximation numérique. Ainsi ce type de tâche :

> déterminer une valeur approchée décimale à 10^{-p} près de l'unique solution d'une équation mise sous la forme f(x) = c, où f est une fonction numérique reconnue bijective et strictement monotone d'un intervalle I sur un intervalle J contenant c. (Birebent 2001, p. 156)

Il montre comment les techniques de résolution de ce type de tâche dépendent de la présence des instruments informatiques comme le tableur de la calculatrice :

> - Actuellement, la méthode du balayage décimal est la plus répandue. Elle a supplanté celle de la dichotomie ce qui nous prouve que l'évolution des techniques, attribuable au perfectionnement des matériels, participe à éloigner les pratiques institutionnalisées, dans les manuels et dans les classes, des intentions noosphériennes initiales.
> - Par rapport à la dichotomie, *le balayage […] est plus rapide à mettre en œuvre sur les calculatrices, maintenant que celles-ci sont munies d'une commande de tabulation des valeurs d'une fonction*.[2] (Birebent 2001, p. 157)

Ainsi le recours à un programme déjà installé sur la machine (la calculatrice ou l'ordinateur) peut fournir directement les résultats effectifs attendus et donc concurrencer l'exécution de tout autre algorithme de prise de valeurs. Pour ce qui est de la programmation de ces algorithmes, l'auteur peut alors souligner, comme Lê Van, sa difficulté à vivre :

> La technique s'est dissoute dans une gestuelle sur machine d'où la notion de convergence est absente ou à peine évoquée, où la formalisation de l'algorithme a laissé place à un programme préfabriqué duplicable sur l'une ou l'autre des calculatrices les plus en vogue sur le marché scolaire, où l'encadrement ne connaît pas de traitement algébrique et ne subsiste que dans une ostension numérique ou graphique. (Birebent 2001, p. 159)

Birebent parle alors d'un phénomène « *d'effacement de la nature algorithmique des procédés d'approximation successifs* » dans les manuels à l'époque, dû à l'arrivée des instruments informatiques :

> Dans leur manière de traiter la tâche, les manuels Belin (1995) et Hatier (1995) ne laissent subsister de l'algorithme que le calcul des premiers termes et des kits de programmation sur calculatrice ou ordinateur. (Birebent 2001, p. 159)

Et il conclut que cette tâche, au cours de son institutionnalisation :

> s'est débarrassée de la mise en forme algorithmique qu'elle a remplacée par une succession de procédures attachées uniquement à la calculatrice. (Birebent 2001, p. 171)

De ces deux études, on peut déduire que, dans la période des années 1990, en France, des algorithmes sont bien présents dans le domaine de l'analyse mais que leur vie est réduite car la responsabilité des calculs numériques est transférée aux instruments de calcul sans engager la construction de l'algorithme. De même, la programmation est quasi-inexistante, puisqu'elle se ramène à l'exécution de programmes préfabriqués dans les calculatrices ou les tableurs.

[1] C'est nous qui soulignons
[2] C'est nous qui soulignons

I.1.3. Nguyen Chi Thanh et Annie Bessot (2003)

Dans leur recherche Nguyen et Bessot questionnent la présence des notions informatiques de boucle et de variable au sein de l'enseignement des mathématiques. L'examen des manuels en 2nde et 1ère des programmes 2000 leur permet d'affirmer la présence d'algorithmes dans le domaine de l'analyse, associés à deux types de tâches :

> Dans le domaine de l'analyse (2nde et 1ère), la notion d'algorithme (dans les 3 manuel examinés) est associée à deux types de tâche : décomposer une fonction donnée en opérations élémentaires et calculer les valeurs numériques d'une fonction. (Nguyen et Bessot 2003, p. 16)

Ils confirment les résultats de Birebent (2001) sur le rôle joué par les instruments de calculs, en particulier des tableurs :

> Or, pour calculer les valeurs numériques d'une fonction, les trois manuels font appel aux tableurs (calculatrice et logiciel). Cet usage décharge l'élève de la répétition des calculs et masque l'intervalle sur lequel on fait ce calcul, et qui, cependant, conditionne son début et sa fin. (ibidem)

Ils étudient alors plus précisément comment sont présentes dans le domaine de l'Analyse la programmation et la notion de boucle.

> *L'écriture d'un programme n'est pas à la charge de l'élève* […]. Par contre un élève de seconde ou de 1ère S doit savoir exécuter un programme donné sur sa calculatrice ou l'expliquer […] Par conséquent, dans les manuels étudiés, *l'écriture d'une boucle ou d'un test d'arrêt n'est pas à la charge de l'élève*. (ibidem)

Le constat est encore plus pessimiste sur la vie de la notion de variable informatique qui, pour exister, doit se différencier de la notion de variable en mathématique :

> La rareté des explications et l'inexistence des exercices, concernant la notion de variable dans un programme informatique, ne peuvent être sans conséquence sur la conception de cette notion. De quels moyens dispose l'élève pour différencier la notion de variable informatique (qui, rappelons-le, désigne une case de la mémoire) et la notion préconstruite (et déjà présente au collège) de variable en mathématique ? (Nguyen et Bessot 2003, p. 16)

Nguyen et Bessot construisent et expérimentent dans deux classes du lycée (2nde et 1e S) une situation didactique dont l'enjeu est la formulation de la répétition d'une action (écriture d'une boucle) dans un programme. Dans cette situation, des binômes d'élèves doivent écrire un message à un robot fictif, « CALCULATOR ». La solution optimale du problème que doivent résoudre les élèves est l'écriture d'un programme avec boucle. Cette écriture engage certains élèves, mais quelques uns seulement, dans un long processus que les auteurs décrivent comme suit :

> Le message final est l'aboutissement d'un long processus au cours duquel les élèves différencient la désignation de la variable de celle de ses valeurs successives et explicitent :
> - la relation entre la « position » de la valeur et l'entier qui multiplie le pas ;
> - la condition d'arrêt (selon le schéma action/test) ;
> - l'initialisation de la variable ; (Nguyen et Bessot 2003, p. 28)

Cette expérimentation conduit Nguyen et Bessot à s'interroger sur les conditions permettant de donner du sens à l'activité de programmation d'algorithmes pour la résolution d'un problème mathématique :

> Cependant, l'expérimentation a montré les limites de cette situation, qui ne fait pas rentrer *naturellement* les élèves dans une problématique de la programmation. En effet CALCULATOR n'est pas une *machine* qui fait effectivement les calculs demandés, valide ou invalide un message codifié. Des interventions (de l'enseignant ou des observateurs) ont été nécessaires pour que CALCULATOR ne soit pas assimilé à un élève (comme le suggérait d'ailleurs l'énoncé).
> Comment transformer cette situation pour qu'elle intègre des conditions permettant de donner du sens à l'activité de programmation des algorithmes dans la résolution d'un problème (*a priori* décidable) ? (Nguyen et Bessot 2003, p. 30)

La caractérisation de ces conditions et la transformation du récepteur des messages des élèves, CALCULATOR, en une « machine » est l'un des problèmes que nous essayons d'aborder tout au long de notre travail de thèse.

I.2. La vie des algorithmes et de la programmation dans EMS en France dans l'habitat de l'arithmétique (Laetitia Ravel 2003)

Le programme de 1998 réintroduit l'arithmétique en Terminale S en manifestant la volonté de développer les aspects algorithmiques. Cette volonté est affichée dès l'introduction des programmes :

> L'objectif est de donner aux élèves un minimum cohérent de notions élémentaires permettant l'élaboration d'algorithmes simples et fondamentaux […]

Ravel (2003) analyse les programmes et les manuels. Elle constate que :

> Depuis sa réintroduction en 1998, en spécialité mathématique, trois fonctions sont assignées à l'arithmétique : mettre en avant l'aspect algorithmique des mathématiques par le biais des principaux algorithmes arithmétiques, permettre un travail « rigoureux » sur le raisonnement, et enfin, sensibiliser à l'histoire des mathématiques. (Ravel 2003, p. 37)

Mais l'auteure montre que la volonté d'introduire une démarche algorithmique dans l'arithmétique n'est que rarement mise en oeuvre dans les manuels et que la vie de cette démarche est là aussi très difficile, comme le montre la réédition de certains manuels :

> […] La volonté de mettre en avant l'aspect algorithmique de l'arithmétique n'est pas relayée par les manuels de 1998. […] Par ailleurs l'usage fait dans les manuels de l'outil informatique dans le cadre des cours n'est pas, dans la plupart des cas, inséré un projet didactique permettant un travail sur la démarche algorithmique (à l'exception notable du manuel Transmaths). Par conséquent, l'intégration de ces outils dans les manuels se révèle difficilement viable, comme le montrent les importantes modifications qui portent sur ce sujet dans les manuels édités en 2002, à l'occasion de programme d'arithmétique. Cette difficulté de faire vivre l'aspect algorithmique de l'arithmétique par le biais de l'utilisation de calculatrice, de tableurs ou d'autres moyens informatiques se retrouve dans le choix de cours des enseignants. (Ravel 2003, p. 266)

L'analyse des pratiques de deux enseignantes lui permet d'identifier un système de contraintes qui pourrait expliquer l'échec de cette tentative :

> Tout d'abord, mettre en avant l'aspect algorithmique est en rupture avec les représentations dominantes de ce qu'est l'activité mathématique dans l'institution scolaire française […]. Par ailleurs, de nombreuses contraintes d'ordre matériel freinent l'utilisation des outils informatiques dans les classes. Enfin il manque des références et des habitudes pour la mise en place d'un enseignement d'« algorithmique » de l'arithmétique. (Ravel 2003, p. 266)

La diversité des outils informatiques pouvant être employés - calculatrice, logiciels de calcul formel ou tableur - pour l'activité de programmation est potentiellement grande :

> Par ailleurs, si la majorité des contenus d'arithmétique peuvent donner lieu à un travail de programmation, les outils informatiques choisis pour ce travail peuvent être très différents. Il peut s'agir de la calculatrice ou d'ordinateur avec l'utilisation soit d'un logiciel de calcul formel, soi un tableur. (Ravel 2003, p. 62)

Mais aussi bien les manuels que les pratiques laissent voir une propension à minimiser la place de ces outils :

> Les différentes éditions des manuels montrent une tendance à diminuer la place accordée à cet outil dans les cours d'arithmétique. Les exercices à résoudre « Avec ordinateur » disparaissent, mais les T.P. de programmation qui sont conservés permettent d'aborder un travail sur les algorithmes par le biais d'une formalisation préalable des algorithmes à programmer pour pouvoir ensuite les traduire en algorithmes de programmation. (Ravel 2003, p. 75)

Ravel conclut à la vie difficile (encore une fois) des algorithmes et de leur programmation, des programmes tout faits étant les plus souvent « montrés » aux élèves. On retrouve dans le domaine de l'arithmétique les conclusions des auteurs précédents : le plus souvent, il ne reste à un élève de Terminal S que la responsabilité de savoir exécuter un programme donné sur sa calculatrice ou de l'expliquer.

I.3. La vie des algorithmes et de la programmation dans EMS au Viêt-nam (Lê Van 2001)

Au Viêt-nam, avant les années 2000, « l'enseignement des mathématiques du lycée au Viêt-nam est marqué essentiellement par la réforme des années 90 » (Lê Van 2001). Contrairement à la France, l'introduction des instruments informatiques, en particulier de la calculatrice, n'est pas préconisée dans l'enseignement des mathématiques et les instruments de calculs traditionnels comme les tables numériques restent dans la salle de classe :

> Comme lors de la contre-réforme [en France], l'impact de l'informatique dans les mathématiques est pris en compte. Cependant, les calculatrices ne sont pas considérées comme un instrument favorisant l'approche de l'informatique ; l'introduction officielle des calculatrices dans l'enseignement des mathématiques n'est pas préconisée. Les outils de calcul traditionnels (tables de logarithmes, tables trigonométriques, …) restent présents. (Lê Van 2001, p. 224)

Contrairement à la France, la tendance est d'introduire des éléments d'algorithmique dans une discipline « Informatique », séparée des mathématiques.

> L'approche de l'informatique passe essentiellement par un enseignement de certains éléments de l'algorithmique, des notions sur l'architecture et les principes de fonctionnement des ordinateurs. De plus, il y a une tendance forte que cet enseignement s'effectue dans une discipline spéciale intitulée « Informatique ». (ibidem)

Les noosphériens vietnamiens, avec lesquels Lê Van s'est entretenu, sont convaincus que, si le point de vue algorithmique relève bien des mathématiques et de l'informatique, l'informatique doit occuper une place prépondérante dans l'initiation à ce point de vue.

> L'aspect algorithmique est considéré comme fondamental à la fois pour les mathématiques et pour l'informatique. Cependant, un rapport à l'algorithmique s'établit essentiellement par une initiation à l'informatique, les mathématiques n'étant considérées dans l'enseignement que comme un terrain pour une approche implicite des algorithmes, sans connaissance spécifique. (ibidem)

Ces mêmes noosphériens caractérisent comme suit les types de tâches qui relèvent de l'algorithmique :

> Exécution d'un algorithme :
> (1) Effectuer des opérations selon un ordre déterminé, conformément à un algorithme donné.
> Construire des algorithmes :
> (2) Dissocier une opération en sous opérations pouvant être effectuées selon un ordre précisé.
> (3) Décrire exactement le processus permettant d'exécuter une activité.
> (4) Généraliser une activité sur certains objets isolés en une activité sur une classe d'objets.
> (5) Comparer différents algorithmes de résolution d'un problème pour déterminer l'algorithme optimal (Lê Van 2001, p. 207)

Lê Van compare les idéologies en œuvre derrière la réforme 1990 au Viêt-nam à celles qui ont conduit à la réforme dite des mathématiques modernes, puis à la contre-réforme en France :

> Nous avons mis en évidence certaines caractéristiques fondamentales des mathématiques qui témoignent des choix épistémologiques pris pour la réforme des années 1990. Ces caractéristiques ne sont ni celles de la réforme des mathématiques modernes en France, ni celles de la contre-réforme. Les mathématiques ne sont pas conçues comme univers de structures définies axiomatiquement. L'aspect expérimental n'est pas considéré comme un caractère fondamental des mathématiques. Cela conduit à promouvoir dans

> l'enseignement l'aspect *méthodique*[1] de l'univers mathématiques basé sur des règles de raisonnement. (Lê Van 2001, p. 224)

Le raisonnement semble donc premier pour les initiateurs de la réforme de 1990 et s'oppose à un travail explicite sur les algorithmes.
S'agissant de la rubrique des manuels de Seconde intitulée « Certains éléments des méthodes et technique de calcul » qui pourrait permettre l'introduction des notions d'algorithme et de programmation, Lê Van met en évidence trois fortes contraintes :

> - Contrainte du temps didactique : le nombre d'heures fixé par le programme n'est pas suffisant pour l'introduction des contenus de ce thème.
> - Contrainte des relations trophiques : ce thème apparaît comme un « isolat » sans lien véritable avec d'autres parties du programme de mathématiques.
> - Contrainte du choix didactique ; le point de vue dominant de la noosphère est d'introduire l'enseignement de l'informatique comme une nouvelle discipline. (Le Van 2001, p. 211)

On peut donc affirmer qu'avant 2000, la vie des algorithmes reste marginale et que l'algorithmique et la programmation sont absentes de EMS au Viêt-nam.

En guise de conclusion

Ces recherches montrent que dans EMS en France (jusqu'à la fin des années 1990) la présence des algorithmes et des programmes informatiques dans les manuels (et donc dans les pratiques effectives, si on considère un manuel comme un modèle de ces pratiques), a tendance à s'amenuiser.
Dans l'habitat de l'analyse et concernant la programmation avec boucle, il ne semble subsister que l'algorithme itératif associé à la suite $x_{n+1} = f(x_n)$, et l'algorithme associé à la méthode « balayage », pour l'approximation des racines d'une équation. Pour ce dernier algorithme, Birebent (2001) pose la question de l'usage des tableurs des calculatrices et l'ordinateur en tant que substitut de cette méthode.
Dans l'habitat de l'arithmétique en Terminale S de spécialité, Ravel (2001) montre l'évincement dans la pratique des enseignants de l'aspect algorithmique au profit du raisonnement. La question de l'existence réelle de la programmation de ces algorithmes dans EMS peut être posée.
En ce qui concerne EMS au Viêt-nam, Lê Van (2001) montre que les calculatrices ne sont pas officiellement présentes contrairement à la France, que la notion d'algorithme ne vit que de façon implicite et enfin que la programmation est complètement absente.

Qu'en est-il en France dans les programmes 2000, les plus récents avec lesquels nous avons travaillés ? Comment les objets informatiques élémentaires comme boucle et variable sont-ils considérés dans ces programmes ?
En France et au Vietnam, dans les programmes actuels, quel rôle jouent les instruments informatiques usuels comme calculatrices et ordinateurs ? Comment ces instruments sont-ils mis en relation avec des algorithmes présents ? Quels sont les algorithmes qui survivent ?
Nous répondrons à cette question dans la partie B de notre thèse.

Nous allons maintenant présenter des recherches qui ont porté sur des tentatives d'introduction d'éléments de programmation dans une option informatique en France dans les années 1980.

[1] Souligné par nous

II. Introduction d'objets de l'informatique *en informatique*

Des études en didactique de l'informatique sont été essentiellement menées dans les années 80, parenthèse dans l'enseignement français : durant une courte période, la création d'une option « Informatique » a manifesté la volonté noosphérienne d'introduire un enseignement de la programmation au lycée, hors du domaine des mathématiques.
Rogalski (1985), Rouchier, Samurçay (1985) cherchent à repérer certaines difficultés dans l'acquisition de notions fondamentales en Informatique, en particulier les notions des boucles et de variables. Ils conçoivent, réalisent et analysent des situations expérimentales destinées à l'option informatique en classe de Seconde.
Méjias (1985) sous la direction de Colette Laborde s'intéresse aux difficultés conceptuelles rencontrées dans l'écriture d'algorithmes itératifs par des élèves de collège.
Lagrange (1992) s'interroge sur l'acquisition de données codifiées en informatique.
Dupuis et Guin (1988) conduisent une recherche sur l'apprentissage de la programmation en Logo.

II.1. Janine Rogalski (1985)

Dans sa recherche concernant « les difficultés conceptuelles dans l'acquisition des premières notions informatiques sur la programmation : l'exemple des structures itératives », Rogalski (1985) formule une hypothèse générale sur l'origine des difficultés conceptuelles des élèves :

> les difficultés conceptuelles des élèves dans une tâche de programmation seraient d'autant plus grandes que la structure de la solution informatique s'éloignerait des représentations spontanées et des méthodes de solution « à la main ». (Rogalski 1985, p. 65)

Elle déduit de cette hypothèse une liste d'affirmations au sujet des structures de boucles :

> - la structure « répète/action/jusqu'à/condition » est la plus accessible ;
> - à l'intérieur de cette boucle, l'explicitation de la transformation d'une variable « compteur », qui contrôle le nombre d'exécutions, ne pose pas de problème majeur (dès lors bien entendu que la séquentialité est respectée) ; […]
> - l'intervention des opérations d'action et de test pour la structure de répétition «tant que/condition/faire» exige de mettre en cause le modèle spontané, on doit donc rencontrer des obstacles importants dans son acquisition ;
> - la structure du type « pour i = 1 à n faire », dans laquelle la variation de la variable « compteur » (i) ainsi que son test ne sont pas explicites, va fonctionner d'abord comme une instruction « miracle » pour laquelle la question de l'adéquation au problème à résoudre aura très difficilement du sens. (Rogalski 1985, p. 65)

Rogalski nomme « modèle spontané des élèves » de la notion de boucle le cycle « *on répète une action et on recommence* ». Les analyses de Rogalski permettent de déduire que la boucle « tant que » est moins accessible que les autres boucles car elle demande une anticipation de condition d'arrêt pour la formuler : on doit anticiper la sortie de la boucle avant de répéter des actions. Ces résultats sont en concordance avec ceux d'autres chercheurs de la même époque :

> Ces éléments sont compatibles d'une part avec les analyses faites de séances ultérieures à la séquence didactique (Rouchier, Samurçay et al, 1984) d'autre part avec les résultats obtenus par Soloway (Soloway, Ehrlich et al, 1982) sur des étudiants ; cet auteur constate en effet que la structure « tant que… » pose encore des problèmes aux étudiants « avancés » et que les étudiants ont des difficultés dans la compréhension de la boucle « pour… » qui, selon ses hypothèses, aurait dû être plus facile puisque contrôlée par le processeur. (Rogalski 1985, p. 65)

Elle montre que la représentation (implicite ou explicite) qu'a l'utilisateur du dispositif informatique (DI), influe sur le contrôle de l'exécution de son programme :

> Lors de l'exécution d'un programme l'élève doit donc mettre en relation les effets obtenus avec la signification qu'il attribue au texte du programme, en utilisant les représentations qu'il se fait (même

implicitement) sur la manière dont le dispositif informatique « traite » le texte du programme. (Rogalski 1988, p. 133)

Les observations d'élèves lui permettent d'identifier les représentations possibles du DI. Elle les hiérarchise en 3 grands niveaux :

- Au niveau 0 un programme n'est pas conçu par l'élève comme une séquence d'instructions destinée à faire produire par un dispositif une classe de résultats pour une classe de données (avec les langages dits impératifs). Cela se traduit en particulier par des difficultés importantes à assimiler les contraintes syntaxiques nécessaires à la communication avec le DI. […] ;
- Le niveau 1 correspond à une confusion entre les instructions tournées vers l'ordinateur et les actions de l'utilisateur […] ;
- Le niveau 2 correspond à l'attribution à l'ordinateur des propriétés de compréhension de l'utilisateur ; (Rogalski 1988, p. 134)

II.2. Renan Samurçay (1985)

De son côté, Samurçay conduit des recherches sur les notions de variable et de boucle, objets qu'elle juge au coeur de la programmation:

La notion de variable, au même titre que celle de boucle, intervient comme une notion centrale dans la programmation informatique […].(Samurçay 1985, p. 144)

Samurçay souligne également la singularité de la notion de variable en informatique par rapport à la notion de variable en mathématiques :

- La planification du traitement des variables dans une répétition est de loin l'activité la plus significative dans la conceptualisation des notions de variable et de boucle. C'est surtout dans cette dernière activité que le concept de variable informatique apparaît différent de celui de variable mathématique (symbole représentant un élément non spécifié ou inconnu d'un ensemble), et que l'opération d'affectation (relation asymétrique) diffère de la relation d'égalité (évidemment symétrique).
- Dans le cadre de l'itération […] les variables informatiques sont nécessairement des fonctions au sens mathématique. (ibidem)

Samurçay conçoit une expérimentation dans laquelle on propose aux élèves des programmes incomplets et que les élèves doivent donc compléter. Elle distingue deux types de variables, celles qui « *correspondent à des données explicites du problème* » et celle qui « *interviennent dans la solution informatique* ». Parmi ces dernières, elle montre que, dans certaines conditions, la variable « compteur » est la mieux traitée :

La variable COMPTEUR est toujours mieux traitée que les autres variables lorsqu'elle obéit au modèle canonique : « elle s'initialise à zéro, elle s'incrémente de 1 ». Lorsque la variable qui joue le rôle de compteur ne correspond pas explicitement à ce modèle, les réussites chutent de moitié comme, par exemple, dans la tâche de construction du test d'arrêt. Les variables d'accumulation de résultats intermédiaires sont dans tous les cas, plus difficiles à traiter. (Samurçay 1985, p. 158)

Elle compare les difficultés des élèves relativement aux trois opérations sur les variables que sont : « l'initialisation » « la mise à jour » et « le test d'arrêt » :

- les élèves font une meilleure utilisation et un meilleur traitement des variables institutionnalisées dans l'enseignement (par exemple le compteur) ;
- l'initialisation des variables est une opération cognitive difficile : des deux modalités d'initialisation c'est la lecture qui joue un rôle privilégié ;
- les difficultés présentées par la construction d'une expression booléenne qui intervient dans un test d'arrêt et par la construction d'un corps de boucle sont du même ordre. (Samurçay 1985, p. 159)

L'initialisation est l'opération qui pose le plus de difficulté aux élèves. Elle rapproche cette difficulté de celle rencontrée par les élèves dans les problèmes additifs « trouver l'état initial, la transformation additive et l'état final étant donné » mise en évidence par Vergnaud (1982) :

On note que la tâche d'initialisation est globalement celle qui pose le plus de problème aux élèves. On peut estimer d'ailleurs que les élèves rencontrent ici une classe de problème qui soulève de difficultés

> d'ordre général dans de nombreux autres domaines (Vergnaud, 1982) : faire une hypothèse sur l'état initial d'une variable, connaissant la transformation et l'état final. (Samurçay 1985, p. 158)

Dans une recherche concernant « la planification d'actions dans la situation de programmation », Rouchier et Samurçay (1985) proposent deux critères pour qu'une situation donne du sens à l'activité de programmation :

> - Elle doit faire intervenir un dispositif d'exécution sans lequel il n'y a pas de programmation ;
> - Cette exécution doit être différée pour qu'il y ait une nécessité à élaborer et à exprimer une procédure. (Rouchier et Samurçay 1985, p. 247)

II.3. Béatriz Méjias (1985)

Méjias conçoit une situation expérimentale dans laquelle un élève-programmeur doit écrire un programme qui doit être exécuté par un opérateur humain. Elle montre que l'écriture du programme produit dépend de la représentation qu'a l'élève-programmeur des connaissances de l'opérateur :

> Les représentations que l'élève-programmeur se fait du fonctionnement de l'opérateur jouent évidemment un rôle important dans la prise de conscience que l'opérateur ne sait faire que certaines opérations bien déterminées. Cette prise de conscience n'est pas immédiate et le premier comportement des élèves au but de l'activité consiste à attribuer à l'opérateur les compétences qu'ils possèdent eux-mêmes. (Méjias 1985, p. 123)

Elle propose dans sa thèse une analyse conceptuelle de l'itération dans laquelle elle distingue le corps de l'itération caractérisé par l'invariance de la séquence d'instructions qui la compose. Elle insiste sur le rôle des interactions entre l'« élève-programmeur » et les résultats renvoyés par la machine (qui joue alors le rôle de milieu) :

> La présence ou l'absence de feedbacks sur les productions des élèves a lui aussi joué un rôle important. Comme on pouvait s'y attendre, ceux dont l'origine est extérieure aux élèves (simulation du robot, ou exécution par le micro-ordinateur) ont eu un rôle moteur dans l'évolution des algorithmes construits. Mais la simulation des programmes par des élèves eux-mêmes a été sans effet, car de façon naturelle ils mettaient en œuvre leurs conceptions du fonctionnement défini par la consigne. (Méjias 1985, p. 125)

Méjias met en évidence le rôle positif joué par des contraintes d'un langage de programmation dans l'écriture des boucles :

> Le rôle du langage dans l'évolution des productions des élèves est apparu clairement : le passage des programmes avec simple mention de l'itération sans corps d'itération à une forme plus achevée de l'itération a été rendu possible par les contraintes du langage. (Méjias 1985, p. 104)

II.4. Jean-Baptiste Lagrange (1992)

Pour son travail sur les conceptions des objets informatiques chez le débutant, notamment les « données codifiées » comme une chaîne de caractères, Lagrange se réfère au concept de systèmes de représentation et de traitement (S.R.T.) de Hoc (1987). Il définit d'abord la notion de « plans » comme un S.R.T. particulier :

> Les plans sont, à l'intérieur des S.R.T. des représentations schématiques servant de guide pour l'action : ne prenant pas en compte les détails d'implémentation, ces représentations sont applicables à une classe de problèmes, et peuvent êtres communs à des S.R.T. distinctes. […] (Lagrange 1992, p. 97)

Il met en évidence les difficultés de passage d'un plan à l'écriture d'un programme informatique, puisqu'il faut entrer dans un processus d'adaptation (aux contraintes d'un langage) d'un S.R.T. non informatique à un S.R.T. informatique :

> En passant du plan au programme, le programmeur débutant doit donc remettre en cause certains aspects non informatiques, en adapter d'autres pour les rendre compatibles avec la programmation pour le dispositif. Par différenciation, adaptation de plans issus de S.R.T non informatiques, il intègre des contraintes du langage et leur donne une signification, et ainsi construit les premiers éléments d'un S.R.T informatique. (Lagrange 1992, p. 97)

Il souligne l'importance d'un dispositif, qu'il conçoit comme l'ensemble « machine – langage » comme milieu pour entrer dans ce processus d'adaptation :

> Je pense au contraire important que les débutants soient confrontés au dispositif ordinateur+langage, en programmant dans la perspective d'un programme utilisable et non en fonction d'exigences didactiques comprises par eux comme extérieures à l'activité. (Lagrange 1992, p. 134)

Conclusion et nouvelles questions

Ces recherches apportent des informations précieuses pour nourrir notre travail, en particulier en ce qui concerne les notions de variable et de boucle.
Cependant ces recherches se réfèrent essentiellement à des difficultés liées à l'apprentissage de la notion de variable par l'élève en accordant une place première au point de vue psychologique. A côté d'une interprétation essentiellement psychologique, n'y a-t-il pas des interprétations possibles de ces difficultés en terme épistémologique ou en termes de situation (au sens de Brousseau) ?
Si on en croit Dupuis et Guin (1988) il paraît impossible d'entrer dans une activité de programmation en mathématique sans avoir déjà fait de la programmation :

> Un enseignement sur les fonctions en troisième, utilisant la programmation Logo, a été expérimenté avec les élèves ayant participé au travail décrit ici. Cet enseignement a été reconduit cette année avec des élèves n'ayant reçu qu'une quinzaine d'heures d'enseignement informatique. Il nous a révélé l'impossibilité d'utiliser des activités de programmation dans l'enseignement des mathématiques avec des élèves n'ayant pas atteint un niveau suffisant de programmation. (Dupuis et Guin, 1988, p. 66)

Pour elles, la solution est de faire précéder de telles activités par une « réelle » alphabétisation en informatique :

> Par conséquent, c'est seulement par une réelle alphabétisation en informatique que l'on peut aborder le problème de l'aide apportée par les activités de programmation dans l'enseignement des mathématiques. (Dupuis et Guin, 1988, p. 66)

Les recherches que nous avons évoquées portent justement sur « l'alphabétisation informatique »[1], c'est-à-dire sur la toute première initiation à l'informatique. Et les élèves, auprès desquels les expérimentations ont été conduites, connaissent au moins des bribes d'un langage de programmation (au minimum 15 heures de programmation). Ce n'est donc pas vraiment la première fois que ces élèves rencontrent les notions de variable et de boucle, et donc de programme informatique dans un langage du style impératif comme Pascal ou Basic. Or, l'introduction de structures répétitives dans un langage évolué avant les expérimentations, peut avoir des conséquences sur l'écriture de boucles comme le montre Méjias (1985). Autrement dit, des interprétations des difficultés d'ordre didactique doivent être aussi envisagées.

La machine est évoquée dans ces recherches : dispositif informatique chez Rogalski (1985), dispositif d'exécution chez Rouchier et Samurçay (1986), dispositif programmable chez Rouchier et al. (1987), machine ordinateur chez Méjias (1985), dispositif ordinateur et langage chez Lagrange (1992). Elles soulignent d'une manière ou d'une autre l'importance de la représentation qu'a l'élève programmeur de la machine, dans les activités de la programmation. Cependant la notion de machine n'est pas explicitée. La relation entre la machine et la genèse des notions de variable ou de boucles n'est pas analysée en profondeur. La question de la co-genèse de la machine et de la programmation n'est pas soulevée.

[1] Pour reprendre un terme introduit par Rogalski en 1985,

Dans les chapitres suivants, en prolongement des études déjà mentionnées au chapitre I nous allons examiner dans les programmes et les manuels de l'enseignement des mathématiques la présence des objets fondamentaux de l'algorithmique et de la programmation, que sont les notions d'algorithme, de programmation, de variable et de boucle.

Chapitre 2
Présence des notions d'algorithme, de boucles et de variables dans l'enseignement des mathématiques au Lycée

I. Analyse des programmes d'enseignement des mathématiques en France

Nous allons conduire une analyse écologique et comparative des programmes des années 1970, 1980, 1990 et 2000. Pourquoi ce choix ? Chacune de ces années marque des changements qualitatifs dans les contenus des programmes de mathématiques relativement à la notion d'algorithme. La fin des années 1960 inaugure la période dite des mathématiques modernes. Cette réforme est brusquement remise en cause en 1980 et la contre-réforme s'appuie sur une modification radicale de l'idéologie des concepteurs de programmes sur ce que doivent être les mathématiques enseignées.

> La contre-réforme en France manifeste une rupture fondamentale par rapport à la réforme bourbakiste des années 1970 : les mathématiques ne sont plus conçues comme un univers de structures, mais comme un champ de résolution de problèmes. L'une des conséquences pour l'enseignement est de mettre en avant le caractère expérimental des mathématiques et la nécessité d'articuler deux moments, expérimental et théorique, considérés comme distincts. Le moment expérimental réserve à l'activité de l'élève une grande place en accord avec les théories constructivistes de l'époque pour lesquelles cette activité est le moteur de l'apprentissage. (Bessot, Comiti 2005, traduit par nous)

Notre analyse se centre sur l'existence des objets fondamentaux de l'algorithmique et de la programmation, que sont les notions d'algorithme, de programme, de variable et de boucle.

Nous limitons cette analyse au programme de mathématiques à option scientifique. Avec elle, nous cherchons à répondre aux questions suivantes :

- Quel habitat occupe la notion d'algorithme ?
- Quelles sont ses niches ?
- Les notions de boucle, de condition d'arrêt et de variable sont-elles présentes dans les programmes ? Comment ?
- Quelles sont, à ce sujet, les tendances de l'évolution des programmes et les raisons de ces évolutions ?

I.1. Programmes des années 1970, 1980 et 1990

Nous résumons ici les principaux résultats mentionnés lors de la revue des recherches en didactique des mathématiques dans ce domaine (cf. Chapitre 1, partie A).

I.1.1. Programme des années 1970

Comme nous venons de le dire, le programme des années 70 est marqué par la réforme des mathématiques modernes. L'accent est mis sur des exposés strictement axiomatiques, les problèmes de fondements, les notions de structures ensemblistes (celles de groupes, d'anneaux et de corps).
En ce qui concerne la notion d'algorithme, elle est introduite implicitement à l'aide des organigrammes qui est le seul langage de description des algorithmes. Cette introduction est faite en relation avec d'autres savoirs comme fonctions, résolution exacte d'une équation ou d'un système linéaire. Un algorithme sert, via un organigramme, à décrire un procédé permettant de calculer ou de trouver des solutions exactes d'une équation et de discuter l'existence de ces solutions. Le rôle de l'algorithme pour le raisonnement est alors mis en

relief. Les algorithmes présents n'étant pas itératifs, les notions de variable et de boucle sont absentes.
Des éléments d'arithmétique sont présents à cette époque avec un point de vue structurel. Ce n'est pas l'algorithme d'Euclide qui est mis en avant comme méthode de recherche d'un PGCD. De la même façon, la mise hors programme de la formule des accroissements finis, et l'absence de la notion de suite font que les méthodes itératives d'approximation d'une équation ne peuvent pas exister. La vie de la notion d'algorithme semble alors fragile. Parmi les manuels étudiés de l'époque (collections Cossart, Riche, Queysanne-Revuz, Durrande), seul Durrande traite en Seconde la notion d'itération à propos de l'usage d'une calculatrice.

L'usage des tables numériques, de la règle à calcul figure dans le programme. Et aussi celui des calculatrices. Cependant l'accent est mis essentiellement sur les deux premiers outils de calcul. Les calculatrices, appelées à l'époque machines de bureau, semblent écartées de la classe à cause de leur coût trop élevé.

I.1.2. Programme des années 1980

Dans ce programme, marqué par la contre-réforme, l'importance accordée au thème des suites en relation avec l'étude de fonctions, la place des activités réservées aux élèves et l'introduction officielle des calculatrices dans l'enseignement des mathématiques s'accompagnent d'une augmentation de la place importante accordée à la notion d'algorithme. L'intention d'introduire l'enseignement de la programmation est officiellement explicitée.

> Développement de la formation personnelle : sûreté dans le maniement des grands moyens de communication : expression écrite et orale, techniques de représentation (schémas, graphiques, dessin industriel), de codage (organigrammes, diagrammes), langage de programmation.

Le domaine de suites numériques est considéré comme un premier terrain pour l'introduction d'éléments algorithmiques. La programmation des suites est introduite dès la classe de Seconde, par exemple pour le type de tâche : « approcher la solution positive de l'équation $x^2 = p$ ». En Première et en Terminale, l'enseignement de la programmation est intensifié avec l'arrivée de méthodes algorithmiques comme la dichotomie, la méthode de la tangente, la méthode Newton.
Ainsi la notion d'algorithme doit outiller, dans le domaine de l'analyse, les techniques des types de tâche « approximation d'un nombre réel » et « approximation d'une solution réelle d'une équation ». L'apparition des calculatrices dans les établissements scolaires doit permettre de valoriser les activités portant sur les algorithmes pour conjecturer, contrôler et comparer les résultats. La notion d'algorithme n'est pas abordée comme un objet de savoir à enseigner mais comme un outil idéal pour, d'abord appuyer l'ambition de développer les démarches scientifiques chez l'élève, ensuite illustrer l'intérêt d'utiliser une calculatrice. L'aspect algorithmique devient présent dans le programme en même temps que l'aspect structurel du programme précédent disparaît.
À cette époque, la plupart des calculatrices des élèves ne sont pas programmables. Ainsi la validation de l'écriture des algorithmes ne peut relever de son implémentation dans la machine de l'élève, mais de la réalisation « étape par étape » avec l'aide d'une calculatrice, ce que certains enseignants appellent « à la main » par opposition à une exécution programmée et automatique. Donc, bien que les algorithmes mentionnés soient souvent itératifs, les notions de boucle et de variable sont absentes du programme. Cet ensemble de conditions pose la question de la viabilité de la programmation des algorithmes dans les manuels. L'examen des collections Colin et Hachette montre que les résolutions approchées associées à certaines méthodes d'approximation (Dichotomie, Interpolation linéaire, Itération) apparaissent aussi bien dans le manuel de Seconde que dans celui de Première (cf. Lê Van

2001) sous forme des méthodes algorithmiques mais que le passage à une programmation de ces méthodes n'est pratiquement jamais abordé dans ces manuels.

I.1.3. Programme des années 1990

Ce programme reprend pour l'essentiel les objectifs et la substance des programmes précédents concernant la notion d'algorithme et ses applications.
Parmi les méthodes d'approximation d'une racine d'équation, il ne reste que deux méthodes, dichotomie et balayage, introduites en Première. Cependant la dernière est préférée à la première car elle est plus facile à mettre en œuvre grâce aux tableurs de la calculatrice (ou de l'ordinateur). La programmation de cet algorithme n'est alors plus nécessaire. Cependant le programme encourage l'introduction d'activités de programmation à tous les niveaux du lycée :

> - En classes de Seconde et de Première, programmer le calcul de valeurs numériques d'une fonction sur des exemples simples.
> - En Terminale, programmer le calcul des valeurs d'une fonction, le calcul des termes d'une suite récurrente.
> - À partir de la classe de Première, savoir programmer une instruction séquentielle ou conditionnelle,
> - En classe de Terminale, savoir programmer une instruction itérative comportant éventuellement un test d'arrêt. (Programme 1990)

La notion de boucle connaît ainsi sa première apparition officielle au niveau de la 1ère S et de la Terminale S. Mais aucune rubrique des programmes n'est réservée à l'enseignement de la programmation : comment cela peut-il se réaliser dans les manuels ? L'examen des manuels de ce programme (collection Mathématiques, Maths) montre que la responsabilité laissée à l'élève est la plupart du temps d'exécuter les programmes ou d'expliquer les algorithmes donnés. Programmer ou construire un algorithme n'est *a priori* du ressort de l'élève.
Concernant les algorithmes en arithmétique (programme 1998), le programme suggère de mettre en place des « notions élémentaires permettant l'élaboration d'algorithmes simples et fondamentaux ». Mais cette suggestion n'est pas suivie par les manuels.

En guise de résumé

On peut repérer deux types principaux d'algorithmes dans EMS dans ces périodes : ceux de la résolution approchée des équations et ceux du calcul des valeurs d'une fonction ou d'une expression. Ils occupent un habitat en analyse, celui des suites et des fonctions. À partir du des années 90, commence une évolution qui va repousser dans les niveaux supérieurs (en Terminale) l'enseignement des algorithmes de résolution approchée et restreindre leur nombre. Le programme 1998-1999 se démarque des précédents par l'apparition des algorithmes en arithmétique dans l'enseignement de spécialité en classe de Terminale S.
Les études précédentes (cf. chapitre A.1) montrent que malgré les recommandations des programmes pour la présence des algorithmes et des programmes informatiques dans les manuels, il n'y a pas d'éléments algorithmiques et de programmation des algorithmes dans les manuels.

I.2. Programme des années 2000

I.2.1. Classe de Seconde

Une nouvelle orientation sur la place de l'outil informatique dans l'enseignement des mathématiques au lycée se fait jour :

> - L'informatique, devenue aujourd'hui absolument incontournable, permet de rechercher et d'observer des lois expérimentales dans les deux champs naturels d'application interne des mathématiques : les nombres et les figures du plan et de l'espace.
> - Il est ainsi nécessaire de familiariser le plus tôt possible les élèves avec certains logiciels.

- On exploitera les possibilités offertes par les tableurs, par les grapheurs et par les logiciels de géométrie.

Cette fois-ci, le programme met en avant des applications de l'informatique au-delà de l'usage des calculatrices en recommandant des logiciels sur ordinateurs.
En classe de Seconde, l'élève doit rencontrer la simulation à l'aide du générateur aléatoire d'une calculatrice ou d'un tableur.

> Un des apports majeurs de l'informatique réside aussi dans la puissance de simulation des ordinateurs ; la simulation est ainsi devenue une pratique scientifique majeure : une approche en est proposée dans le chapitre statistique.

Depuis la classe de quatrième, l'élève a été initié à l'usage des tableurs et doit, en seconde, prolonger cet usage à la statistique. L'usage d'autres logiciels est fortement recommandé :

> En Seconde l'usage de logiciels de géométrie est indispensable

Ainsi l'aspect algorithmique est implicitement présent, à travers le tableur, dans l'habitat des fonctions avec les notions d'étapes de calcul et d'enchaînement :

> Sur le tableur, explicitation des différentes étapes du calcul d'une formule en appliquant d'une colonne à l'autre une seule opération (+, −, ×, /, carré, √). Explication de l'enchaînement des fonctions conduisant de x à f(x).

La recherche des solutions approchées d'une équation, habitat privilégié des algorithmes dans les programmes précédents prend un caractère imprécis : aucune méthode de calcul n'est nommée à ce niveau.

I.2.2. Classe de Première

Ce programme affirme le caractère mathématique de certaines notions comme boucle, test :

> Certaines notions informatiques élémentaires (boucle, test, récursivité, tri, cheminement dans les graphes, opérations sur des types logiques) font partie du champ des mathématiques et pourraient être objets d'enseignement dans cette discipline.

Il insiste sur l'obligation de mettre en œuvre ces notions sur la calculatrice :

> Compte tenu de l'horaire imparti et des débats en cours, il n'est proposé aucun chapitre informatique. Néanmoins, l'élève devra mettre en œuvre, notamment sur sa calculatrice, les notions de boucle et test.

On notera que les méthodes d'approximation d'une solution, qui était un habitat privilégié de la notion d'algorithme dans les programmes précédents, mobilisent maintenant à la fois la calculatrice et le tableur.

> L'approximation souhaitée d'une solution sera déterminée par une table de valeurs obtenue sur la calculatrice (en réduisant le pas d'un facteur 10 à chaque étape).

Il en est de même pour la programmation des suites. Dans le programme précédent, c'est la programmation sur la calculatrice qui servait à calculer des termes d'une suite. Mais le programme 2000 écrit : « calcul des termes d'une suite sur calculatrice ou tableur ».
On peut craindre que l'utilisation des logiciels de calculatrice ou d'ordinateur ne rentre en conflit avec l'ambition d'introduire les objets algorithmiques comme boucle.
En effet, certaines calculatrices actuelles fournissent des valeurs d'une fonction ou d'une suite numérique grâce à des commandes pré-établies sans obligation de construction d'un programme à partir d'un algorithme : nous nommerons tableur-calculatrice un tel tableau. Un tableur-logiciel (comme Excel) demande, lui, que l'on formule des procédures de calcul : il relève d'une problématique de la programmation en style fonctionnel dans laquelle les notions de boucle et de variables (comme case d'une mémoire) ne sont pas pertinentes.

I.2.3. Classe de Terminale

Dans le domaine de l'analyse, des algorithmes sont présents sous le nom de méthodes. Ce sont les méthodes d'approximation pour la résolution des équations. Le programme en recommandant l'usage de la calculatrice ou du tableur, comme en 1ère S, et en limitant les ambitions mathématiques « *L'étude de suites $u_{n+1} = f(u_n)$ pour approcher une solution de l'équation f(x) = x n'est pas un objectif du programme* » s'oppose à l'introduction d'un point de vue algorithmique sur ces méthodes :

> On pourra approcher la solution de l'équation f(x) = k par dichotomie ou balayage avec la calculatrice ou au tableur ;
> L'étude de suites $u_{n+1} = f(u_n)$ pour approcher une solution de l'équation f(x) = x n'est pas un objectif du programme : la dichotomie, le balayage suffisent au niveau de la terminale pour des problèmes nécessaires de telles approximations.

I.3. Conclusion

I.3.1. En ce qui concerne les habitats des algorithmes

En analyse, fonctions et équations sont des habitats assez stables de la notion d'algorithme. À propos des fonctions, l'algorithme concernant le calcul des valeurs numériques d'une fonction (problème de tabulation d'une fonction) est potentiellement présent à toutes les époques étudiées.
Pour l'habitat des équations, dans le programme des années 1970, les algorithmes (non itératifs) concernent seulement la résolution « exacte » d'une équation (du 1er ou du 2e degré) ou d'un système d'équations. Par contre, dans les programmes des années 80 et 90 apparaissent des algorithmes de nature itérative concernant la résolution approchée d'une équation à l'aide des suites. Cependant à partir des programmes des années 90, commence une évolution qui va repousser dans les niveaux supérieurs (Terminale en 2000) l'enseignement des algorithmes de résolution approchée et restreindre leur nombre. Il reste seulement deux méthodes : dichotomie et balayage introduite en terminale S.
L'habitat des suites pour la notion d'algorithme reste stable à partir des programmes des années 1980.
Le programme de l'année 2000 se démarque des précédents par la création d'un nouvel habitat, celui de l'arithmétique dans l'enseignement de spécialité en classe de Terminale S (en vigueur depuis l'année scolaire 1998-1999).

Nous mettons ainsi en évidence des techniques de nature algorithmique (nommés méthodes ou non nommés dans les programmes) permettant d'accomplir un certain nombre de types de tâches mathématiques :

En analyse

- « calculer des valeurs d'une fonction » ;
- « calculer des termes d'une suite » ;
- « approcher des solutions d'une équation » (méthodes dichotomie ou balayage) ;
- « approcher une fonction » (méthode d'Euler).

En statistique et probabilité

- « simuler des tirages, des lancers d'un dé »

Les programmes de 2000 autorisent donc à ces praxies d'exister dans les manuels du programme 2000. Il reste à questionner leur existence dans la réalité de l'enseignement.

I.3.2. En ce qui concerne l'outil informatique

L'usage d'un nouvel outil informatique, le tableur, aux trois niveaux du lycée est préconisé dans les programmes :

- Sur tableur, explicitation des différentes étapes du calcul d'une formule (classe Seconde) ;
- Calcul des termes d'une suite sur calculatrice ou tableur (classe Première) ;
- On pourra approcher la solution de l'équation f(x) = k par dichotomie ou balayage avec la calculatrice ou au tableur (classe de Terminale)

Avec des calculatrices de plus en plus performantes, une nouvelle niche apparaît dans le programme 2000, celle de l'utilisation des calculatrices pour « réaliser des programmes ».

> On veillera à faire réaliser sur la calculatrice des programmes où interviennent boucle et test.

La notion d'algorithme sert ainsi à introduire non seulement quelques « notions informatiques élémentaires (boucle, test, récursivité etc.) » mais aussi à renforcer l'utilisation des outils informatiques.

> Il ne s'agit pas d'apprendre à devenir expert dans l'utilisation de tel ou tel logiciel, mais de connaître la nature des questions susceptibles d'être illustrées ou résolues grâce à l'ordinateur ou la calculatrice et de savoir comment analyser les réponses fournies.

1.3.3. Nouvelles questions

Cette analyse nous permet de nous poser de nouvelles questions pour entreprendre l'analyse des manuels :
Comment la notion d'algorithme vit-elle effectivement dans les habitats repérés ? Quels sont les langages pour sa programmation dans les manuels ? Dans quels habitats vit la programmation ? Comment les manuels interprètent-ils les types de tâche « Construire un algorithme » ou « programmer une instruction séquentielle, conditionnelle et itérative » présents dans programme ? Comment l'introduction de la notion de boucle en tant qu'objet de savoir est-elle mise en œuvre dans les manuels ? Quelle est la place des logiciels, en l'occurrence le tableur, dans ces activités ? La notion de variable, indispensable à la conception de la notion de boucle, est absente du texte du programme. Or ce dernier insiste sur la nécessité d'introduire la notion de boucle et de test indissociable de la notion de variable. La notion de variable est-elle avancée par les manuels ? Si oui, comment ?

II. Résultats de l'analyse des manuels en France (programme 2000)

Dans cette partie, nous nous contenterons de donner les grandes lignes de notre analyse institutionnelle ; par moments nous renvoyons le lecteur pour plus de détail aux annexes.

II.1. Partie identifiable au « cours »

L'analyse des programmes 2000 a montré qu'une niche possible de la notion d'algorithme est de permettre l'étude de notions de programmation comme boucle avec condition d'arrêt et variable. Comment cela est-il réalisé dans les manuels examinés ?

II.1.1. Deux types d'algorithmes

L'examen des manuels permet de dégager deux catégories de praxies pouvant être liées soit à un algorithme non itératif, soit à un algorithme itératif.

- algorithme non itératif :
  - +) « calcul de valeur d'une expression »
  - +) « calcul des solutions exactes d'une équation »
- algorithme itératif :

+) « calcul des termes d'une suite » ;
+) « calcul des valeurs d'une fonction » ;
+) « calcul approchée des solutions d'une équation » (dichotomie et balayage) ;
+) « approcher une fonction : méthode d'Euler » ;
+) « simuler des tirages, des lancers de dé ».

Cette typologie montre que la principale niche des algorithmes est le calcul numérique effectif.
Notre intérêt pour la notion de boucle explique que nous nous centrerons par la suite sur les praxies associées à un algorithme itératif. Le nouvel habitat des algorithmes en statistique et en probabilité, repéré dans les programmes, semble exister dans les manuels, mais nous ne l'analyserons pas ici.

II.1.2. Absence de l'enseignement d'éléments d'algorithmique

Dans la partie strictement réservée au cours des manuels examinés, tous les algorithmes sont « montrés » (souvent sous la forme d'un organigramme ou d'un programme écrit en « langage calculatrice ») dans des problèmes résolus ou dans les travaux dirigés.
En classe de Seconde, dans le chapitre des fonctions, les types de taches « décomposer une fonction donnée en opérations élémentaires » ou « calculer les valeurs numériques d'une fonction » sont associés à de nouveaux objets : « calculogrammes » (Déclic), « montages des fonctions » (Belin) ou « boîtes noires » (Bréal). Ces objets pourraient permettre d'introduire la comparaison des algorithmes. Par exemple, pour un calcul des valeurs de « $x^2 + 3x + 2$ », différents schémas (ou différents algorithmes) sont « économiquement » dissemblables en termes de coût ou de nombre d'opérations. Cette piste n'est jamais suivie. Il s'agit simplement de montrer un algorithme dans la partie cours que l'élève devra reprendre dans la partie exercices.
De plus, en Terminale, comme le nombre des méthodes de résolution approchée est réduit à deux, à savoir les méthodes dichotomie et balayage, et qu'il n'est envisagé aucune comparaison entre ces deux méthodes, ni d'évaluation d'une méthode, nous pouvons conclure que l'enseignement d'éléments algorithmique en général et la comparaison des algorithmes en particulier a pratiquement disparu de l'enseignement des mathématiques secondaires (mais a-t-elle jamais vraiment existé ?).

II.1.3. Présence implicite de la notion de variable

La notion de boucle est introduite dans la programmation des termes d'une suite (notamment dans les manuels Belin et Déclic) à l'aide d'un organigramme. Cependant les opérations concernant les variables dans les programmes informatiques des manuels examinés sont en général écrites sans mentionner explicitement la notion de variable. Belin et Déclic accompagnent les programmes de commentaires concernant les instructions. Nous avons essayé de repérer les diverses explications portant sur l'affectation d'une variable. Dans un même manuel, par exemple Déclic, les fonctions du symbole « $\rightarrow$ » peuvent être le calcul, le stockage ou l'initialisation. Belin s'appuie sur l'égalité de 2 nombres pour expliquer le symbole « $\rightarrow$ » : « If C = 1 ; 0 $\rightarrow$ A : Si C = 1, A = 0 ».
Dans ce dernier manuel, deux types d'opérations à l'intérieur d'une boucle, sont induits par le symbole « $\rightarrow$ » :
• La mise à jour : Int(rand×2) + S $\rightarrow$ S ; N + 1 $\rightarrow$ N ; I + 1 $\rightarrow$ I
• L'initialisation : F $\rightarrow$ L2(A) ; L1(A) + 1 $\rightarrow$ L1(A), etc.
La mise à jour porte essentiellement sur la variable « compteur » et obéit au modèle canonique « s'initialise à zéro et s'incrémente de 1 ».

La notion de test d'arrêt est quasiment implicite : les programmes s'arrêtent quand le résultat est obtenu.
Parmi les différents commentaires donnés par les manuels à la notation d'affectation la mémoire (de la calculatrice) est évoquée comme le montre l'exemple ci-après.

Pour introduire la notation d'affectation, le premier exemple de programme porte sur le problème « cet entier est-il premier ? » (Déclic 2002, Seconde, p. 19) :

On entre le nombre N et on va le diviser successivement par D, qui prend les valeurs 3, 5, 7, …, de 2 en 2	Input « Nombre », N $1 \rightarrow D$ Lbl 1 $D + 2 \rightarrow D$ […]

Cependant la définition de la notion de variable en relation avec la notion de mémoire n'est jamais faite par ces manuels.

II.1.4. Absence de la prise en compte de types de boucles

En statistique et en probabilité, les programmes sont exclusivement écrits avec la boucle « pour » alors que dans le domaine de l'analyse, les programmes peuvent faire intervenir tantôt la boucle « aller à », tantôt la boucle « tant que ». Même dans le cas où l'usage de la boucle « tant que » est possible, la boucle « aller à » est utilisée. L'omniprésence des organigrammes dans des manuels constitue l'une des raisons de l'usage prédominant de cette boucle comme nous le verrons plus loin.
À propos de langage, programme en langage calculatrice et organigramme sont les deux langages algorithmiques toujours présents. Dans certains manuels, peut apparaître le triplet (organigramme, programme en langage des calculatrices, commentaires sur le programme). Les algorithmes et les programmes étant donnés tout faits, l'organigramme a le rôle d'illustration ou de schématisation de la structure algorithmique correspondante.

II.1.5. Usage prédominant des tableurs au détriment de la programmation des algorithmes

Rappelons que le programme des années 90 introduit pour la première fois des activités de programmation concernant le calcul des termes d'une suite. Les études précédentes (cf. chapitre A1) montrent que cette directive n'est pas respectée dans des manuels et par les enseignants. Le programme des années 2000 propose moins d'activités de programmation mais se veut plus précis : « *l'élève devra mettre en œuvre, notamment sur sa calculatrice, les notions de boucle et de test* ». Cependant nous découvrons que tous les manuels programment aussi ces algorithmes sur le tableur Excel. De plus, la calculatrice à l'aide de la touche « Ans » permet avec une certaine facilité, de produire des termes de suites définies par récurrence (cf. Annexe A2).
De plus pour les rares algorithmes de calcul approché des équations en Terminale cette pratique de passer par un tableur se confirme. Dans les deux algorithmes, dichotomie et balayage, Déclic programme le premier sur la calculatrice et le deuxième sur le tableur. Nous avons montré (cf. Annexe A2) que la programmation sur tableur-ordinateur de l'algorithme balayage est beaucoup plus facile que celle sur calculatrice : la programmation sur tableur évite la construction de boucle. Ceci pourrait être l'une des explications de la présence dominante, dans des manuels, de la méthode balayage par rapport à la dichotomie.
La vie réelle des notions de programmation comme boucle, variable s'avère donc en deçà de celle que laisse imaginer les programmes.

II.2. Analyse des exercices

Comment les manuels interprètent-ils les types de tâche « Construire un algorithme » ou « programmer une instruction séquentielle, conditionnelle et itérative » présents dans le programme ?
L'examen de trois manuels 2000 (Belin, Déclic, Bréal) nous permet de repérer 8 types de tâches associés à des algorithmes, les trois derniers types concernant plus particulièrement la notion de variable :

T1 : Écrire un algorithme pour un problème donné
T2 : Exécuter un programme donné
T3 : Trouver le résultat d'un algorithme donné pour différentes instances d'un problème
T4 : Expliquer un programme ou un algorithme
T5 : Traduire un algorithme donné dans un langage de programmation
T6 : Contrôler des valeurs particulières d'une variable
T7 : Mettre à jour une variable
T8 : Modifier le test d'arrêt d'une boucle

Type de tâches	**T1**		**T2**		**T3**		**T4**		**T5**		**T6**		**T7**		**T8**	
Manuels	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**	**2°**	**1S**
Belin	0	1	2	5	22	4	0	0	0	0	0	1	0	1	0	1
Déclic	2	0	2	5	15	0	1	0	0	0	0	1	0	1	0	0
Bréal	0	0	4	1	5	1	5	2	3	1	0	0	0	0	0	0

Tableau 1. Nombre d'exercices de type Ti selon les manuels et selon les niveaux

Le type de tâche « Construire un algorithme » se réduit à « Écrire un algorithme pour un problème donné » (T1) et il est pratiquement inexistant (voir tableau 1).
Le type de tâche « programmer une instruction séquentielle, conditionnelle et itérative » se retrouve dans soit dans T1 « Écrire un algorithme pour un problème donné » soit dans T5 « Traduire un algorithme donné dans un langage de programmation ». Le tableau 1 montre que seul Bréal propose des activités.
On note que les exercices sur la notion de variable informatique (T6, T7 et T8) sont quasi inexistants (tableau 1).
Seuls les exercices de type T3 (en seconde) et dans une moindre mesure du type T2 sont bien représentés, tout particulièrement au niveau seconde dans Belin (22) et Déclic (15). Les types T4 (7) et T5 (4) apparaissent comme spécifiques au manuel Bréal : ce manuel offre ainsi la possibilité de travailler sur des algorithmes donnés par les questions posées : « *que fait l'algorithme ?* » ou « *traduire l'algorithme dans un langage de calculatrice et exécuter l'algorithme* ».

Conclusion

En considérant les manuels comme un premier modèle des pratiques enseignantes, cette analyse permet de formuler certaines remarques sur la place et le rôle des algorithmes et de la programmation des algorithmes dans l'enseignement des mathématiques secondaires.

La niche principale des algorithmes est clairement de fournir des résultats numériques exacts ou approchés. Dans cette niche, que nous pouvons qualifier d'algorithmique numérique, les pratiques liées à l'usage des calculatrices et des tableurs entravent fortement la transformation des algorithmes et d'éléments de programmation en objets d'enseignement.

L'écriture d'un programme n'est pas à la charge de l'élève (absence de savoir sur la construction des algorithmes, rareté des exercices T1). Par contre l'élève du lycée doit savoir exécuter un programme donné sur sa calculatrice ou l'expliquer (importance des exercices T2 et T3). Par conséquent, l'écriture d'une boucle ou d'un test d'arrêt n'est pas à la charge de l'élève.

Dans le domaine de l'analyse, malgré les recommandations des programmes sur la mise en œuvre des notions de boucle et de test, on peut constater, dans ces 3 manuels, la prédominance de la programmation sur tableurs-ordinateurs des algorithmes avec boucle. Cet usage décharge l'élève de l'itération des calculs et masque le test d'arrêt, comme par exemple l'intervalle sur lequel on fait ce calcul lors de calcul des valeurs d'une fonction. Nous faisons l'hypothèse que cet usage rend difficile la formulation de la répétition des calculs par l'écriture d'une boucle. En particulier, il efface la nécessité de la formulation de l'initialisation et de la condition d'arrêt de cette action répétée.

La rareté des explications et l'absence d'exercices concernant la notion de variable dans un programme informatique, ne peuvent être sans conséquence sur la conception de cette notion. Or cette notion est intrinsèquement liée à la construction de l'invariant d'une boucle, notamment dans la mise à jour. De quels moyens dispose l'élève pour différencier la notion de variable informatique (qui, rappelons-le, désigne une case de la mémoire) et la notion préconstruite (et déjà présente au collège) de variable en mathématique ? Or l'étude de Samurçay (cf. chapitre A.1.) montre que la prise en compte de l'itération pourrait permettre cette différenciation.

Chapitre A3
La tentative d'introduction de l'informatique dans EMS du Lycée au Viêt-nam

Introduction

L'enseignement secondaire au Viêt-nam depuis 1975, année de l'unification du pays, est essentiellement marqué par la réforme des années 1990[1].
Des éléments d'informatique comme algorithmes et programmation n'existent pas dans les programmes de mathématiques avant 1990. Ces éléments ne sont introduits qu'à partir de cette réforme marquée par une tentative d'introduction de l'informatique dans l'enseignement secondaire.
Cette introduction est tentée à deux endroits : en mathématiques et en informatique laquelle est considérée comme discipline autonome. Cette tentative concilie ainsi deux points de vue opposés des noosphériens comme le montre Lê Van (2001) :

> De plus, cela [l'introduction de l'informatique] peut se faire à la fois à travers l'enseignement de l'informatique en tant que discipline spéciale et à travers l'enseignement d'autres disciplines traditionnelles, tout spécialement l'enseignement des mathématiques. (Lê Van 2001, p. 207)

Dans ce chapitre, nous analysons l'introduction d'éléments d'informatique dans EMS en nous centrant (comme pour l'analyse en France) sur les algorithmes, les langages de programmation, les notions de variable et boucle.

- Quel habitat occupe la notion d'algorithme ? Quelles sont ses niches ?
- Quels sont les types d'algorithmes présents ? Quel type de description est adopté ?
- Les notions de variable et de boucle sont-elles présentes dans les programmes ? Comment vivent-elles dans les manuels ?

I. Analyse du programme de la classe 10

L'informatique est introduite lors de la réforme des années 90 au seul niveau de la classe de la classe10 (équivalente de la classe de Seconde en France). Le chapitre dans lequel cette introduction est faite s'intitule « Éléments sur les méthodes et techniques de calcul » et se place comme dernier chapitre de la rubrique « algèbre » comme le montre le tableau 1.

Algèbre (2,5 séances/semaines sur 82 séances au total)	Géométrie (2,5 séances/semaines sur 82 séances au total)
I. Fonction et graphique (20 séances) II. Équations, système d'équation et inéquation (42) III. Éléments sur méthodes et techniques de calcul (15) Révision, contrôle (5)	I. Vecteurs (13 séances) II. Relation métrique dans triangle, cercle (18) III. Déplacements et similitudes (11) Révision, contrôle (11)

Tableau 1. Contenus du programme des années 90 de la classe 10 (Traduit par nous)

Le chapitre « Éléments sur les méthodes et techniques de calcul » comporte 4 rubriques :

1. Information et représentation des informations. Code. Sommaire sur langages de programmation ;
2. Algorithme. Organigrammes. Propriétés. Types d'algorithmes. Exemples de programmation ;

[1] Une nouvelle reforme est prévue depuis l'année 2002. Le nouveau programme est actuellement en expérimentation dans une cinquantaine de lycées au Viêt-nam.

3. Sommaire sur l'ordinateur, de son histoire et de son rôle ;
4. Notion de méthode numérique. Quelques méthodes numériques simples. (Traduit par nous)

Les commentaires du programme placent clairement le chapitre « Éléments sur les méthodes et techniques de calcul » du côté de l'informatique :

> Les connaissances introduites dans ce chapitre visent à ouvrir aux élèves sur un champ d'application plus vaste : une familiarisation préalable à l'ordinateur et l'informatique. Précisément, l'objectif est de présenter d'une manière sommaire ce qu'est l'information et la représentation de l'information, le codage et le langage, un algorithme et sa programmation, quelques principes du fonctionnement de l'ordinateur. Cet enseignement vise à fournir aux élèves dès connaissances initiales sur la représentation, l'organisation et le traitement de l'information codée, structurée et algorithmique. […]
> La méthode de présentation consiste à utiliser des exemples concrets issus du programme mathématique pour introduire des types et des propriétés fondamentales de la notion d'algorithme qui est la notion centrale du chapitre. Quant à la programmation, il suffit de présenter un exemple très simple en langage le plus utilisé. Enfin, il faut organiser une visite d'une salle d'ordinateur. (p. 20) (Traduit par nous)

De plus, parmi les objectifs de l'enseignement de l'algèbre en classe 10, il est précisé que « *l'élève doit avoir des connaissances initiales sur la notion d'algorithme et sur le calcul automatique* ».
Cependant si le rôle de la notion d'algorithme semble occuper une place centrale, la programmation reste secondaire dans les objectifs de ce programme. Cette place est d'autant plus réduite que l'on peut noter l'absence de toute référence dans le programme aux notions de boucle et de variable. Qu'en est-il dans les manuels ?

Les seuls algorithmes itératifs introduits le sont en tant que méthodes numériques mais on peut constater l'absence totale des instruments de calcul dans cette introduction, hors l'ordinateur. L'ordinateur semble être évoqué seulement dans le but de faire comprendre ses principes de fonctionnement et non pas comme instrument de calcul effectif. On peut prévoir que les seules situations d'introduction des notions d'algorithmes et de programmation sont des présentations « orale » ou « écrite » sans calculs effectifs instrumentés, sauf à la main. Ceci semble être en contradiction avec ce que recommande les promoteurs du programme :

> A l'heure actuelle, plusieurs méthodes numériques peuvent être exécutées à l'aide de l'ordinateur, contrairement au temps auparavant où ceci n'était faisable que sur le plan théorique (car le calcul suivant ces méthodes sans l'ordinateur prendrait des milliers d'années) […] L'introduction de cet enseignement permet à l'élève à initier à des méthodes et techniques de calculs et d'instruments de calcul les plus modernes. (Traduit par nous)

Que se passe-t-il dans les manuels ?

II. Analyse des manuels

À l'époque il y a trois collections de manuels rédigés par trois équipes : les rédacteurs en chef sont respectivement Phan Đức Chính, Trần Văn Hạo, et Ngô Thúc Lanh[1]. Nous nommons chacun de ces trois manuels M1, M2 et M3.
Nous résumons ici les principaux résultats de l'analyse entreprise sur ces trois manuels (cf. Annexe chapitre A3).

[1] Les auteurs des manuels de la classe 10e (en vigueur entre 1990 et 1998) en algèbre de ces trois collections sont : Ngô Thúc Lanh, Vũ Tuấn, Trần Anh Bảo ; Phan Đức Chính, Ngô Hữu Dũng, Hàn Liên Hải ; Trần Văn Hạo, Phan Trương Dần, Hoàng Mạnh Đệ, Trần Thành Minh.

II.1. Partie cours

II.1.1. La structure de cours varie d'un manuel à l'autre.

Le programme désigne les contenus officiels concernant l'enseignement : algorithme, programmation, ordinateur et méthodes numériques. On peut noter une variation dans la structuration des cours concernant ces contenus.

• Dans le manuel M1, l'ordre dans lequel les contenus sont introduits est le suivant : calcul approché, notion d'algorithme, algorithme et calcul automatique, ordinateur. Ce manuel accorde de l'importance à la notion d'erreur (calcul approché) et à l'aspect théorique de la notion d'algorithme qui est défini ; des propriétés (séquentialité, compréhension, déterministe et exactitude, finitude, universalité) sont données et enfin l'algorithme d'Euclide est démontré.

• De son côté, le manuel M2 présente les contenus du programme dans l'ordre suivant : notion d'algorithme, modèle de fonctionnement de l'ordinateur, langage des organigrammes, erreurs et algorithmes d'approximation d'un nombre, histoire des instruments de calcul. Il établit un parallèle entre l'exécution d'un algorithme sur l'ordinateur et le fonctionnement d'un ordinateur. Les notions de variable et de boucle sont introduites (nous verrons plus loin comment). Dans ce manuel, on trouve des méthodes numériques (comme la méthode du point fixe) après l'introduction de la notion d'algorithme et en relation avec celle-ci.

• Le manuel M3 aborde les contenus du programme dans l'ordre suivant : méthode numérique, algorithme, machine ordinateur, informatique. Deux types de méthodes numériques sont introduits : calcul approché d'un nombre (méthode du point fixe) et calcul des valeurs numériques d'un polynôme suivant le schéma de Hörner. Cependant ces méthodes sont abordées avant la notion d'algorithme et sans relation avec celui-ci.

II.1.2. Trois points de vue différents concernant le calcul approché

A propos des méthodes numériques, en conformité avec le programme, ces trois manuels introduisent tous le calcul approché concernant la racine (carrée ou cubique) d'un nombre mais d'une manière différente d'un manuel à l'autre.
Dans M1 on ne trouve que des formules comme :

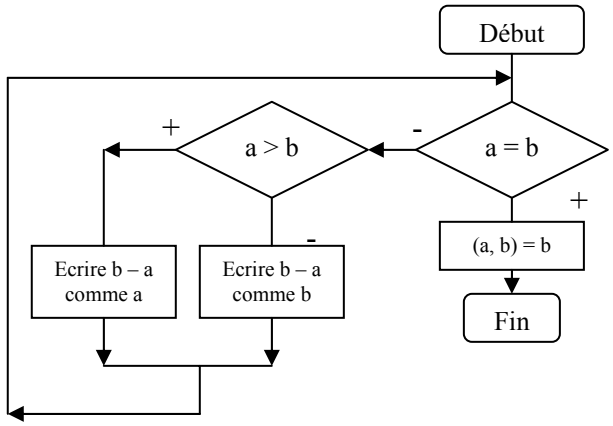
$$\frac{1}{1+\alpha} \approx 1-\alpha ; \frac{1}{1-\alpha} \approx 1+\alpha ; \sqrt{1+a} \approx 1+\frac{\alpha}{2}, \sqrt[3]{1+\alpha} \approx 1+\frac{\alpha}{3}$$ (M1, p. 138)

M2 est le seul à décrire l'algorithme de calcul approché en langage organigramme. De son côté, en présentant (sans démonstration théorique) la méthode du point fixe, M3 introduit la suite défini par récurrence $x_{n+1} = \varphi(x_n)$. Ceci amène à obtenir une valeur approchée de $\sqrt{2}$ en choisissant $x_1 = 1{,}4$ et en calculant $x_2 = 0{,}5(x_1 + 2/x_1)$ etc. Cette méthode est introduite sans relation avec la notion d'algorithme. Aucun instrument de calcul n'est mentionné. Le point de vue algorithmique n'est pas alors mis en valeur à travers ces calculs ce qui est en contradiction de l'intention du programme.

II.1.3. Présence du langage des organigrammes et de la description algorithmique avec numérotation de ligne et absence de langage algorithmique structuré

Les trois manuels suivent les recommandations du programme en ce qui concerne les langages algorithmiques. Les algorithmes présents sont d'emblée représentés par une description avec numérotation des étapes. Puis les organigrammes sont introduits avec des règles bien précises sur les pavés d'actions, de test etc. Par exemple, l'algorithme d'Euclide est représenté selon ces deux descriptions comme suit (M1, p. 148) :

Organigramme



Algorithme

DEBUT

Etape 1 : Si a = b alors (a, b) = (a, a) = a. Fin.
Sinon aller à l'étape 2.
Etape 2 : Déterminer le plus grand nombre parmi a et b.
Etape 3 : Remplacer le plus grand nombre par la différence entre le plus grand et le plus petit.
Etape 4 : Retourner à l'étape 1.

FIN.

Figure 1. Description de l'algorithme d'Euclide selon deux descriptions, organigramme et étape par étape (M1)

La raison de l'introduction des organigrammes n'est en général pas explicitée sauf dans le manuel M3 :

> Après avoir trouvé un algorithme pour résoudre le problème il faut le représenter de telle sorte qu'une autre personne puisse le comprendre et l'appliquer.
> En général, on représente un algorithme par un dessin sous forme un schéma. […]
> L'organigramme présente un grand avantage, il donne une image claire et panoramique des opérations de l'algorithme. (M3, p. 151) (Traduit par nous)

Sont mises en avant une fonction de communication et une fonction d'aide à la compréhension des organigrammes.
La notion d'algorithme est définie dans les trois manuels. Un algorithme est une suite d'indications (M1) ou d'instruction (M2) *pour résoudre un problème*, alors que pour M3 il est un système de règles pour aboutir, en un nombre fini d'étapes, au résultat escompté.

> Les indications pour réaliser des opérations, les actions dont on a parlé précédemment sont appelées algorithme pour résoudre le problème posé. (M1, p. 141)
> Une liste finie d'instructions à suivre pas à pas afin de résoudre un problème quelconque est appelée algorithme de la résolution de ce problème. (M2, p. 112)
> Un algorithme est un système rigoureux et exact de règles déterminant l'ordre dans lequel des opérations sont exécutées de telle sorte qu'après un nombre fini d'étapes on obtienne un résultat escompté. (M3, p. 151) (Traduit par nous)

Seul le manuel M2 introduit dans la définition d'un algorithme la notion d'instruction.

II.1.4. Quasi absence d'une définition des notions de variable et de boucle

Dans ces trois manuels, c'est l'algorithme d'Euclide qui sert à montrer la notion de boucle.
Les notions de variable et de boucle ne sont pas explicitées dans le manuel M1, mais sont implicites comme le montre la figure 1 : dans la description de l'algorithme, la notation d'affectation est outillée par des expressions comme « *écrire b – a comme b* » ou « *remplacer le plus grand par la différence entre le plus grand et le plus petit* ».
La notion de variable n'est mentionnée à aucun moment dans M3. Par contre la notation d'affectation est introduite en affectant la valeur d'une expression à une autre expression (alors qu'en général on affecte une valeur à une variable).

> La notation « := » exprime l'opération de l'affectation. Cette opération est exécutée de la manière suivante : calculer l'expression se trouvant à droite puis affecter sa valeur à l'expression se trouvant à gauche. (M3, livre d'exercices, p. 153) (Traduit par nous)

La confusion entre écriture mathématique et opération d'affectation montre les conséquences de l'évitement de la notion de variable dans le manuel M3 :

$$i := i + 1,\ \ x_i := \frac{1}{2}\left(x_{i-1} + \frac{1}{x_{i-1}} \right)\ (M\,3,\ \ livre\ \ d'exercices\ \ ,\ p.\,160\,)$$

La variable x_i ne peut être affectée dans un programme que par le recours à une variable intermédiaire.
Le manuel M2 est le seul à expliciter la notion de variable. Cette notation est d'abord définie comme « *une lettre ou une chaîne de caractère utilisée comme nom pour une grandeur qui se transforme* » puis elle explicitée en relation avec la mémoire au cours de l'évocation de l'exécution d'un calcul répétitif sur l'ordinateur :

> Dans notre modèle de l'ordinateur, chaque variable est associée à une boîte de stockage. Une étiquette du nom de variable est collée sur la boîte. Dans la boîte se trouve un papier sur lequel est écrit la *présente valeur* de la variable*. La variable est le nom de la valeur se trouvant dans la boîte.* Chaque boîte possède un couvercle que l'on peut ouvrir pour affecter une nouvelle valeur à la variable. De plus, il y une fenêtre grâce à quoi on peut lire cette valeur sans la modifier. (M2, p. 120) (Traduit par nous)

Pour M2 « *la variable est le nom de la valeur se trouvant dans la boîte* ». Cette explicitation, malgré la relation avec la mémoire, ne fait par la distinction entre la valeur d'une variable à un moment du processus de calcul et le nom associé à cette variable.
La notion d'affectation et sa notation par le signe « ← » est introduite de façon classique :

> Dans la boîte 2, on a une affectation. Plus précisément, cela veut dire : « Affecte à la variable SOMME la valeur de DERNIER + AVANT DERNIER ». La flèche ← est appelée opération d'affectation. (M2, p. 122) (Traduit par nous)

En relation avec un calcul répétitif, deux boucles sont présentées, différenciée par la position du test d'arrêt qui peut être contrairement à M1 placé avant.

a) Répéter la tâche T jusqu'à ce que la condition C soit vraie ;
b) Tant que la condition C est encore vraie alors répéter la tâche T ;

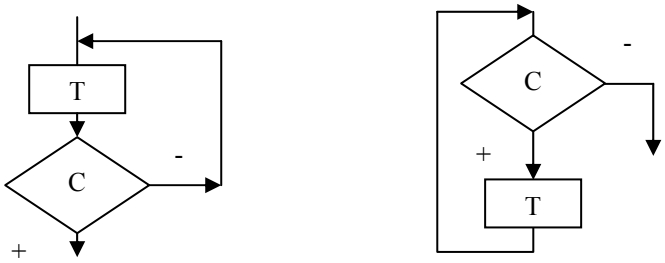


Figure 2. Organigramme des deux boucles du manuel M2

Ces deux structures de boucle ne sont cependant pas utilisées dans la suite, y compris dans l'écriture des algorithmes itératifs.

La machine est évoquée mais pas présente dans l'introduction de la notion d'algorithme
On peut noter que seul M2 introduit des notions des boucles et des variables en relation avec le modèle de fonctionnement d'une machine ordinateur. D'autres manuels présentent toujours la notion d'algorithme avant l'introduction des ordinateurs.

II.2. Partie exercices

L'examen de la partie d'exercice nous permet de déterminer des types de tâches associés à des algorithmes itératifs présents dans les manuels :

+) Calculer le PGCD de deux entiers (algorithme d'Euclide) (Ae) ;
+) Calculer des termes d'une suite Fibonacci (Af);
+) Calculer la somme de nombres pairs (As) ;
+) Calculer la racine carrée (cubique) d'un nombre (Ar) ;
+) Calculer les valeurs d'un polynôme (schéma d'Horner) (Ap) ;
+) Déterminer le plus petit réel parmi n réels donnés (Am) ;

Le tableau 2 décrit la présence de ces algorithmes itératifs dans différents manuels :

Manuel	Ae	Af	As	Ar	Ap	Am
M1	×					
M2	×	×	×	×	×	×
M3	×			×	×	

Tableau 2. Présence des types de tâches selon les manuels M1, M2 et M3

On peut noter que les types de tâches faisant intervenir les algorithmes d'Euclide, du calcul approché d'un nombre et du calcul des valeurs d'un polynôme sont présents dans presque tous les manuels. Or ces algorithmes ne sont pas introduits dans EMS au sein d'organisations mathématiques. Ces algorithmes sont donc des isolats au sein de EMS. De plus les calculs effectifs de ces algorithmes ne sont pas instrumentés en accord avec les programmes mais en désaccord avec ce que recommande les promoteurs de ce programme :

Rappelons les types de tâches concernant des algorithmes repérés au Lycée en France :

T1 : Écrire un algorithme pour un problème donné
T2 : Exécuter un programme donné
T3 : Trouver le résultat d'un algorithme donné pour différentes instances d'un problème
T4 : Expliquer un programme ou un algorithme
T5 : Traduire un algorithme donné dans un langage de programmation
T6 : Contrôler des valeurs particulières d'une variable
T7 : Mettre à jour une variable
T8 : Modifier le test d'arrêt d'une boucle

Sont-ils présents dans les manuels vietnamiens ?

Manuel	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**
M1	2		1					
M2	19	1		2			4	
M3	5		2					

Tableau 3. Nombre d'exercices de type Ti selon les manuels M1, M2 et M3

La notion de variable est pratiquement absente (T6, T7 et T8) puisque seul le manuel M2 propose quatre exercices de mise à jour d'une variable.
L'absence de machine ordinateur et la place secondaire accordée à la programmation se traduisent ici par l'inexistence des types de tâches T2 et T5.
Le type de tâche « écrire un algorithme pour un problème donné » est prédominant (contrairement à la France), tout spécialement dans M2. Notons que 15 sur les 26 algorithmes à écrire sont non itératifs : ce sont les algorithmes de construction géométrique et de résolution exacte d'équations, d'inéquations, de systèmes d'équations linéaires du 1er degré.

Nous pouvons rattacher cette prédominance à trois raisons :
- L'absence d'instruments de calcul qui ne permet pas de produire des calculs effectifs à partir d'un algorithme itératif.
- Parmi les 26 algorithmes à écrire, seulement 2 ne sont pas introduits dans le cours (Ap et Am dans la partie cours de M2), les autres sont, soit déjà introduits dans le chapitre, soit des algorithmes non itératifs mais dont on connaît une solution mathématique comme par exemple la résolution exacte d'une équation ;
- L'idéologie « méthodique », l'une des caractéristiques de l'enseignement vietnamien des mathématiques dans le secondaire, consiste à attendre des élèves qu'ils sachent énoncer des règles, décrire des processus de calcul et de construction de géométrie etc. comme le déclare Lê Van (2001) :

> Nous avons mis en évidence certaines caractéristiques fondamentales des mathématiques qui témoignent des choix épistémologiques pris pour la réforme des années 1990. Ces caractéristiques ne sont ni celles de la réforme des mathématiques modernes en France, ni celles de la contre réforme. Les mathématiques ne sont pas conçues comme un univers de structures définies axiomatiquement. L'aspect expérimental n'est pas considéré comme un caractère fondamental des mathématiques. Cela conduit à promouvoir dans l'enseignement l'aspect méthodique de l'univers mathématique basé sur des règles de raisonnement. (Lê Van 2001, p. 224)

Seul M2 demande l'écriture de d'algorithmes itératifs Ap et Am « nouveaux » (au sens où ils sont non présentées dans la partie cours). Mais aucun élément technologique (au sens de Chevallard) pour les concevoir n'est introduit dans la partie cours (du manuel M2).
Or, en l'absence du contrôle pragmatique de la machine, seuls ces éléments technologiques peuvent amener un contrôle théorique.

III. Conclusion

De notre analyse de l'introduction d'éléments d'informatiques dans EMS nous dégageons plusieurs points.

1. L'objectif central du programme est d'introduire la notion d'algorithme avec définition et propriétés. L'introduction de la programmation joue un rôle secondaire. Aucun commentaire sur celle-ci n'est présent dans le programme. Les notions de variable, de boucle ne sont pas mentionnées.
Le passage par des calculs numériques semble marquer la volonté des initiateurs du programme de favoriser des relations entre la notion d'algorithme et les mathématiques enseignées. Mais aucun problème mathématique de EMS n'exige d'algorithmes itératifs qui seuls donnent du sens à la notion de variable informatique et de boucle. Ceci et l'absence totale de recours à des instruments de calcul (ordinateur et calculatrice) risque de compromettre cette volonté.

2. Les manuels étudiés semblent respecter les intentions du programme en ce qui concerne la notion d'algorithme. Description séquentielle avec numérotation des étapes et organigramme sont les seuls types de langages de représentation des algorithmes. Aucun langage algorithmique structuré n'est introduit. Plusieurs algorithmes non itératifs présents dans le programme de EMS sont introduits. Ces algorithmes sont prédominants par rapport aux algorithmes itératifs qui forment un isolat et ont donc peu d'avenir dans l'enseignement. La prédominance des algorithmes non itératifs va dans le sens du caractère méthodique de EMS au Viêt-nam : ces algorithmes sont considérés avant tout comme des méthodes mathématiques, c'est-à-dire des règles ou des processus à suivre pour la résolution d'une classe de problème.

3. Le manuel M2 se distingue des deux autres de deux façons :

- par rapport à la notion de variable qu'il introduit en relation avec la notion de mémoire dans un problème de calcul répétitif. La notation d'affectation, indispensable dans la distinction de cette notion avec la notion de variable en mathématique, est explicitée.
- par rapport aux calculs numériques qu'il est le seul à présenter comme des algorithmes et qu'il décrit à l'aide d'organigrammes.

Mais ce chapitre disparaît des manuels de mathématique lors de la fusion des 3 collections en une seule collection en l'an 2000 dont le rédacteur en chef est le rédacteur principal du manuel M2. Il est remplacé par un chapitre intitulé « Erreurs » dans lequel les notions d'algorithme et de programmation ne sont plus introduites. Lê Van (2001) a mis en évidence les contraintes qui imposent cette disparition : « contrainte du temps didactique, contrainte des

relations trophiques, contrainte du choix didactique, contrainte de l'instrument de calcul ». D'autres raisons se surajoutent : ce chapitre est placé en fin d'année, ses contenus sont absents des révisions de fin d'année.

Conclusion de la partie A

Les analyses entreprises dans cette partie nous permettent de dégager les points suivants.

1. Il y a bien volonté d'introduire des éléments d'informatique dans les deux institutions, Lycée en France et au Viêt-nam, mais avec deux points de vue différents.
Au Viêt-nam, l'organisation de l'enseignement proposée est d'introduire d'abord la notion d'algorithme en tant qu'objet de savoir avec définition, propriétés, exemples, puis d'évoquer le fonctionnement d'un ordinateur absent.
En France, l'enseignement a la volonté d'attacher des notions de base en informatique (comme boucle, test, structure de données) à des domaines des mathématiques, comme l'analyse, la statistique et récemment l'arithmétique, sans que la notion d'algorithme soit objet d'enseignement.
Dans les deux cas, le topos de l'élève est très réduit comme nous l'avons montré en analysant les exercices proposés aux élèves dans les manuels

2. Dans les deux institutions, on peut faire un constat d'échec de ces tentatives mais à des degrés différents.
Au Viêt-nam, on a assisté à la suppression totale du chapitre informatique dans EMS et une deuxième tentative (introduction de l'informatique en tant que discipline) a, elle aussi, été éphémère (voir annexe A3-1). En France, si les notions de base en informatique se maintiennent, elles vivent mal : cela se traduit dans les manuels par le refus de construire des types de tâches relatives à la programmation des algorithmes et par le recours à des logiciels de calcul pour instrumenter les techniques associées à ces tâches.
Notre analyse confirme donc les visions pessimistes dégagées dans les études précédentes (chapitre A1).

3. On peut avancer certaines raisons explicatives de cet échec.
Au Viêt-nam, le chapitre concerné est un « isolat » dans EMS (pour reprendre des termes de Lê Van 2001), puisque les algorithmes itératifs nécessaires à cette partie sont peu présents dans les mathématiques enseignées. De plus, l'absence ou la non prise en compte des instruments de calcul (calculatrice, ordinateur) font que les algorithmes itératifs de calcul ne sont jamais effectifs au-delà des calculs des premiers termes de l'itération (calculs à la main).

En France, il existe dans EMS des habitats possibles pour les algorithmes itératifs de calcul en analyse, en statistique et en arithmétique. Leur présence a été voulue par les promoteurs de la contre-réforme pour introduire l'activité expérimentale dans l'enseignement des mathématiques. Pour ces noosphériens, une condition pour faire vivre ces algorithmes de calcul, et l'activité expérimentale, est la présence des instruments de calcul. Mais les logiciels comme le tableur (de la calculatrice et l'ordinateur) qui sont préconisés, prennent en charge les calculs itératifs et une part de l'élaboration de ces algorithmes.

Au-delà des différences relevées dans les motivations institutionnelles, les conditions matérielles (équipements informatiques disponibles dans les classes) et les contenus et les organisations didactiques, nous pensons qu'une analyse de nature épistémologique sur les savoirs visés, notamment les notions de variable et de boucle, peut nous permettre d'approfondir les raisons de cet échec.

C'est pourquoi, nous étudions dans la partie suivante (partie B), comment a émergé un enseignement d'algorithmique et de programmation (en même temps qu'a émergé l'informatique comme discipline scientifique) à l'université afin de dégager les conditions mises en place par les pionniers de cet enseignement et les choix possibles d'organisation des savoirs de base de cet enseignement : les stratégies d'enseignement (chapitre B1).

Ces stratégies possibles conduisent à interroger la notion de machine et de langage du point de vue épistémologique. Pour cela, nous cherchons à repérer les ruptures et les filiations dans la co-genèse de la notion de machine ordinateur et de la programmation des algorithmes (chapitre B2) puis les raisons du passage d'un langage machine aux langages évolués (chapitre B3).

Partie B

Algorithmique et la programmation :
une enquête épistémologique

Chapitre B1
Émergence d'un enseignement d'algorithmique et de programmation

Dans ce chapitre, nous nous intéressons à l'émergence d'enseignements universitaires d'algorithmique et de programmation.
Quelques jalons historiques rassemblés dans un préambule nous permettront de justifier le découpage de notre analyse en trois moments.

Préambule

L'enseignement de la programmation précède l'enseignement de l'algorithmique

De même que le mot « ordinateur », crée par Perret en 1955 à la demande de la société IBM (France), le mot informatique est une création française, celle de Dreyfus en 1962. En 1966, ce dernier est reconnu par l'Académie française, mais ne figure pas encore dans le dictionnaire de Robert (cf. Baron 1987). Or le mot « programmation » est présent dans l'enseignement universitaire (au moins en France) depuis la fin des années 50 comme l'atteste la description faite par Kuntzmann de la genèse de l'informatique à l'université de Grenoble :

> Enseignement
> Voici d'abord les principales créations. Elles ne sont pas toutes directement liées à l'informatique au sens strict.
> 1945 Mathématiques appliquées en section spéciale Haute Fréquence
> 1948 Analyse appliquée 1ère année de l'INPG
> 1949 Devient aussi certificat de licence. Section spéciale Mathématiques appliquées à l'INPG (1 an pour ingénieurs déjà diplômés).
> 1957 DEST *Programmation*[1]
> 1959 Réforme de la licence TMP MMP Analyse numérique Logique et Programmation Statistique.
> 1960 Section normale IPG (3 ans)
> 1966 Maîtrise MAF et *Informatique*. Institut de Programmation. (Kuntzmann 1988, souligné par nous)

Ainsi on peut constater que l'enseignement de l'« informatique » débute historiquement par un enseignement de la « programmation ». La mise en place de cet enseignement dans la section spéciale « Mathématiques appliquées » (citée par Kuntzmann) nous permet de penser qu'au début de son développement, l'informatique se cache « dans l'ombre » des mathématiques appliquées, comme l'affirme Baron (1987) :

> L'informatique cependant a ceci de particulier qu'elle n'existait pas comme champ autonome à la fin des années 50, et qu'elle a conquis sa légitimité dans l'ombre d'autres disciplines, au premier rang desquelles les mathématiques. Ce sont donc les recherches effectuées ***aux marges*** de ces disciplines qui ont produit le corpus initial de connaissances savantes nécessaires à la constitution d'une science. (Baron 1987, p. 34)

C'est à travers la programmation que se constituent les premiers contenus de formation de ce qui deviendra plus tard un enseignement d'informatique au début de l'université en France.

L'émergence de l'enseignement de l'algorithmique

Dans les premiers enseignements de programmation, le mot « algorithme » n'est pas encore présent. L'examen des divers rapports du CNRS dans les années 60 permet à Baron de le constater :

[1] C'est nous qui soulignons

> Ainsi le rapport national de conjecture scientifique du CRNS de 1964 ne mentionne le mot « algorithme » qu'à deux reprises, les deux fois en faisant référence au langage Algol qui est présenté comme étant particulièrement adapté : une fois dans le cadre de l'analyse numérique, et une fois sous le titre « langage de programmation ». (Baron 1987, p. 52)

En expliquant la naissance du langage évolué Algol dans les années 60, Moreau montre que la résolution de problèmes technologiques, en particulier la compilation des langages, a permis la prise de conscience que les algorithmes sont l'objet de la communication homme – machine. C'est donc pour lui l'une des conditions principales de l'émergence de l'algorithmique :

> Jusqu'en 1960, l'effort avait principalement porté sur la mise au point d'une traduction efficace des langages de programmation. […] Mais, une fois les problèmes de compilation bien maîtrisés, il était naturel que les théoriciens étudient cette fois, comme ils avaient commencé à le faire avec Algol, la forme que devait avoir la définition même d'un langage. C'est pourquoi, à la fin de la période historique, les jalons ont été posés d'une réflexion sur la programmation. Cette réflexion est liée, non plus à la communication entre l'homme et la machine, mais à l'objet même de cette communication, les algorithmes. (Moreau 1987, p. 188)

Le développement des langages de programmation et la réflexion sur la conception des programmes (preuve, comparaison etc.) donne, à la fin des années 60, naissance à une nouvelle branche de l'informatique, celle de l'algorithmique. Les recherches en l'algorithmique conduisent à l'essor des applications de l'informatique dans la société. Moreau qualifie cette étape de cruciale dans la genèse de la science informatique (distincte des mathématiques) :

> Or cette génération des applications a fait que, en 1963, l'informatique n'était plus l'apanage des mathématiciens. Une nouvelle science venait de naître avec ses domaines de recherche propres, ses applications propres, et une industrie qui l'entraînait ou la suivait. (Moreau 1987, p. 200)

L'enseignement de l'algorithmique va alors se séparer à l'université de celui des langages de programmation :

> Elle [La méthode LCP – Logique de Conception des Programmes] a permis de passer d'un enseignement visant essentiellement l'acquisition de connaissances techniques à un enseignement orienté vers l'apprentissage d'une méthode, en mettant en évidence les étapes fondamentales du processus d'élaboration des programmes ; plus généralement la pratique de la méthode LCP a conduit à dissocier au niveau de la formation l'enseignement de l'algorithmique de celui des langages de programmation. (Ducateau et Fleck, 1993)

Les jalons historiques que nous venons d'évoquer très rapidement nous permettent de faire un choix de moments clés pour mieux repérer l'arrivée dans l'enseignement universitaire des objets informatiques fondamentaux comme boucle et variable, les conditions de leur mise en place et les relations qu'ils peuvent entretenir avec les machines et les langage de programmation :

- 1957 : premier enseignement de programmation mis en place à l'université Joseph Fourier par J. Kuntzmann.
- 1968 : première édition du volume « Fundamental algorithms » dans la série « The art of computer programming » de Knuth.
- 1978 : parution d'un ouvrage de référence sur l'algorithmique « Fundamentals of Computer Algorithms » de Horowitz et Sahni.

Nous considérons ces trois ouvrages comme des traités au sens de Neyret:

> Nous appelons « traité » le livre produit par un auteur (considéré comme institution transpositive), appartenant à la sphère de production du savoir […], ou très proche de celle-ci, dans le but d'élémenter les savoirs à enseigner. (Neyret 1995, p. 57)

Élémenter désigne, pour Neyret et pour nous, le travail de l'auteur d'un traité pour découper et organiser le corps des savoirs à enseigner autour d'objets fondamentaux (et non élémentaires). Ce travail ne revient donc pas à une simplification du savoir mais à sa transposition institutionnelle vers l'université, considérée ici comme une institution d'enseignement.

I. Kuntzmann (1957) « Cours de moyens de calcul – Atelier arithmétique »

Lors de 50e anniversaire de l'informatique à Grenoble, le CNRS salue la contribution essentielle de Jean Kuntzmann à la genèse de cette discipline :

> En 1951 le professeur Jean KUNTZMANN (1912 – 1992) créait officiellement à l'Institut Polytechnique de Grenoble, le premier « laboratoire de calcul ». Cet événement peut être considéré comme point de départ de l'aventure informatique à Grenoble. On connaît son aboutissement actuel : un tissu de plusieurs centaines d'entreprises à la pointe de la technologie, un ensemble très complet de formations académiques de haut niveau et un panel de laboratoires de recherche de renommée internationale. Cet ensemble fait de Grenoble l'un des tout premiers pôles actuels de l'informatique en France. (CRNS, 2002 sur http://www2.cnrs.fr/presse/communique/)

La contribution de Kuntzmann dans la création et l'élaboration de contenus de cette nouvelle discipline universitaire en France est aussi soulignée par Baron :

> Des enseignements *d'analyse numérique*, et de mathématiques appliquées se mettent assez vite en place dans quelques centres universitaires. Ainsi, à INP de Grenoble, sous l'impulsion du mathématicien Jacques KUNTZMANN, des cours d'analyse appliquée sont organisés dès 1948. Ensuite, dans les années 50, des cours d'analyse numérique et de programmation sont mis en place : un DEST de programmation en 1957, un certificat « techniques de programmation » en 1962. Un institut de programmation sera également mis en place en 1966-1967. (Baron 1987, p. 35)

Le cours de Kuntzmann, intitulé « Cours de moyens de calcul – Atelier arithmétique » est destiné à la « formation de Calculateurs ». Ce cours se compose de quatre chapitres répartis dans deux fascicules :

> Chapitre 1. Les machines : I. Vue d'ensemble des divers types d'ateliers ; II. Notion de niveau ; III. État brut et état organisé, le programme ; IV. Les instructions, le programme ; V. Constituants d'un atelier arithmétique ; VI. La mémoire ; Classification des types d'ateliers d'après la mémoire.
> Chapitre 2. Les instructions : I. Les machines à programme normal ; II. L'enchaînement des instructions ; III. Les instructions arithmétiques ; IV. Classification d'après le nombre d'adresse dans les instructions arithmétiques ; V. Aiguillage ; VI. Tests et comparaison ; VII. Modification d'instructions ; VIII. Autres instructions logiques.
> Chapitre 3. Le programme : I. Organigramme ; II. Boucles ; III. Exemples de programmation empruntés aux problèmes commerciaux ; IV. Bibliothèque de programmes ; V. Sous-programme ; VI. Insertion de sous-programmes dans un programme ; VII. Emploi des sous-programmes dans les programmes ; VIII. Bibliothèque de sous-programme ; IX. Application d'une machine idéale dans une machine réelle.
> Chapitre 4. Étude de l'entrée et de la sortie : I. Sortie ; II. Entrée.

Dans cet enseignement dit de programmation, l'auteur établit un lien étroit entre la programmation, l'architecture de la machine, son fonctionnement et l'écriture des programmes à une machine donnée.

Une simple lecture du sommaire de ce cours, enseigné à l'université au début de l'ère informatique, atteste de :

- la présence des notions fondamentales telles que machine et mémoire, boucle, programme, programmation et instructions, organigramme ;
- l'absence de certaines notions concernant la programmation, présentes dans certains cours d'initiation à l'informatique dans l'enseignement universitaire actuel : architecture de Von Neumann, variable, algorithme, langage de programmation, langage de la machine ;

- la présence de notions qui ne sont plus utilisées dans l'enseignement actuel pour une initiation à l'informatique : machines à programmes normales, atelier arithmétique, etc.

Examinons maintenant de plus près comment s'articulent les objets informatiques dans ce cours.

I.1. Langage et programme

La notion de « langage de programmation » est absente de son cours puisqu'elle est inconnue à l'époque. Cependant, Kuntzmann accorde une place centrale à la notion d'instruction, élément considéré actuellement comme base d'un langage de programmation. Ces instructions sont écrites dans un langage dit « conventionnel » dont le dictionnaire (pour son explication en langage ordinaire) est un « code d'instructions ».

> Au niveau des nombres, le calcul à effectuer est indiqué à l'atelier sous forme d'instruction. Chaque instruction est l'indication, dans un langage conventionnel d'une opération simple (somme, produit, transfert, comparaison) […] Le dictionnaire de ce langage conventionnel s'appelle code d'instructions. Il contient la liste de toutes les instructions possibles et l'explicitation en langage ordinaire de leur signification. (Kuntzmann 1957, p. 34, fascicule 1)

Les langages Gamma et EDSAC qui lui servent à illustrer la notion de langage conventionnel ne sont pas distingués par Kuntzmann.

> Par exemple, en langage GAMMA :
> 8 3 0 0 signifie : transférer le nombre qui est en mémoire opérateur dans la mémoire 3 (p. 10, fascicule 1)
> En langage EDSAC : *An :* Ajouter le nombre placé dans la position de mémoire n à celui qui est dans l'accumulateur. La somme reste dans l'accumulateur. (ibidem)

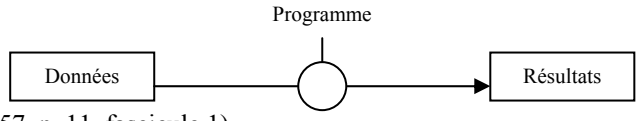
Rappelons que le langage Gamma est considéré actuellement comme un langage machine et le langage EDSAC comme un langage assembleur. Le choix d'un langage est celui du constructeur de la machine :

> Le code d'instructions est choisi par le constructeur et l'utilisateur n'a aucune action sur lui. (Kuntzmann 1957, p. 10, fascicule 1)

Il est alors le langage d'une machine. Il y a donc autant de langages que de machines.

Un programme est pour Kuntzmann « un ensemble organisé d'instructions » :

> Pour résoudre un problème, il faut fournir à l'atelier des données et un ensemble organisé d'instructions. Cet ensemble se nomme le programme.
> On peut schématiser de la manière suivante, les relations entre les données, les résultats et le programme



(Kuntzmann 1957, p. 11, fascicule 1)

La notion de niveau (p. 4, fascicule 1), objet d'enseignement, sous-tend chez Kuntzmann à la fois l'idée d'organisation des calculs et celle des instructions, et est donc organisatrice du travail de programmation. Kuntzmann définit trois niveaux :

- le niveau « *des interprétations mathématiques ou des phrases* » qui est « *propice à la compréhension de l'ensemble d'un calcul* »,
- le niveau « *des opérations et des nombres* » qui est « *propice à la description détaillée de l'exécution d'un calcul* »,
- le niveau « *des chiffres* » où « *on manie les chiffres et les tables réglant les opération sur les nombres d'un seul chiffre* ».

Si la description algorithmique des programmes est absente du cours, par contre on constate la présence systématique d'organigrammes « *schémas traduisant l'état organisé d'un calcul* » (p.5)
La construction d'un organigramme fait entrer nécessairement dans un travail mathématique comme le montre l'auteur dans les deux exemples donnés : interpolation, calcul d'une valeur numérique de polynôme.
À l'organigramme sont attachées des formes graphiques comme flèche, rectangle, triangle, pour spécifier des types d'instructions (calcul, test, sortie etc.), et des règles d'écriture.

> On remarquera que l'organigramme d'un calcul
> - ne peut comporter qu'un départ
> - ne peut se terminer que sur un signal d'arrêt, mais peut comporter plusieurs arrêts différents. (Kuntzmann 1958, p. 2, fascicule 2)

Kuntzmann décrit alors la place indécise de l'organigramme dans l'activité du programmeur. Rappelons qu'à cette période, la conception de l'algorithme du programme et la programmation de cet algorithme sont effectuées par une même personne.
Puis, il prend position pour un usage systématique d'organigrammes en avançant deux raisons :

- c'est un outil permettant la conception et la construction d'un programme ;
- cet outil, en se constituant en objet intermédiaire entre un programme et un problème mathématique, doit aussi permettre une réflexion critique sur un programme.

> Certains programmeurs font d'abord un organigramme, puis écrivent le programme. D'autres se passe d'organigramme ou l'établissent après coup. Il semble, en réalité, que les deux choses doivent aller de pair. L'organigramme est un guide pour l'écriture du programme. Mais inversement, l'écriture du programme réagira sur l'organigramme si l'on cherche des raffinements. L'essentiel est d'avoir en fin de travail, à la fois un programme correct et un organigramme intelligent. Cet organigramme sera essentiel pour étudier les modifications éventuelles du programme. (Kuntzmann 1958, p. 3, fascicule 2)

L'auteur fait reposer le travail de conception ou de correction d'un programme sur des organigrammes.
La volonté de se libérer des contraintes des langages des machines se manifeste donc à travers l'organigramme et la notion de niveau.
Cependant, on attend des élèves « calculateurs » qu'ils écrivent le programme final dans un langage machine. Nous avons vu que chaque machine a un langage : il y a donc nécessairement un choix dans les cours de Kuntzmann sur les machines.
Quelles machines parmi les machines existantes à l'époque, sont présentes dans ce cours ? Pourquoi ? Quelles sont les caractéristiques de ces machines ?

I.2. Machine : effaçabilité de la mémoire et boucle

Dans un chapitre intitulé « Machine », il introduit la notion générique d'« atelier » pour désigner un environnement dans lequel un calcul mathématique s'effectue. Cet environnement peut intégrer ou non une machine.

> Sans vouloir faire une classification, nous citerons :
> - l'atelier manuel n'utilisant que du crayon et du papier ;
> - l'atelier avec machines de bureau ;
> - l'atelier avec machines à cartes perforées ;
> - l'atelier avec grosses machines à relais ou électroniques ;
> - l'atelier composé utilisant plusieurs sortes de matériel. (Kuntzmann 1957, p. 1, fascicule 1)

Cette classification permet à l'auteur de caractériser une machine qui constituerait à elle seule un atelier :

> Une machine à calculer est dite <u>à programme</u> si elle constitue à elle seule un atelier sans avoir besoin d'être complétée par un opérateur humain. Nous admettons pourtant, dans des cas exceptionnels, un tel

> opérateur pourvu que son intervention se borne à une tâche purement matérielle, sans avoir jamais une décision à prendre. (Kuntzmann 1957, p. 28)

Une première caractéristique d'un tel atelier est d'être « arithmétique » et une deuxième d'être « numérique » :

> Nous voulons insister au départ sur le fait qu'un atelier arithmétique résolvant un problème par une méthode donnée […] Il n'est pas inutile d'insister sur cette caractéristique car d'autres atelier ne la possède pas, par exemple des méthodes graphiques ou analogiques […] (Kuntzmann 1957, p. 1, fascicule 1)

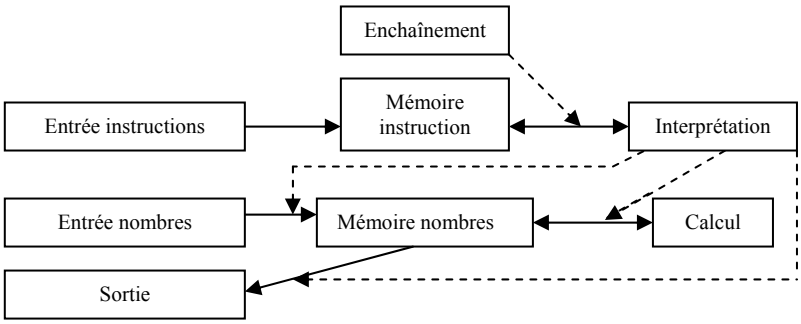
La caractéristique « arithmétique » peut être rapportée aux calculs arithmétiques exécutés au sein de la machine ou aux types de problèmes essentiellement de calcul (arithmétique), résolubles par les machines de l'époque. Ces deux premières caractéristiques sont bien sûr communes à d'autres ateliers.
Une troisième caractéristique est la « vitesse » de calcul qui, elle, va permettre de distinguer entre eux les différents ateliers (à la main, machine à bureau, cartes perforée etc.). Pour Kuntzmann, cette dernière caractéristique conduit l'homme « *à transférer à la machine le travail qui était confié primitivement à l'homme* » et « *à remplacer l'esprit humain par un automate* ». Il aboutit ainsi à la quatrième caractéristique « automatique » d'une machine.
Il donne ensuite les « constituants » de la machine.

> […] Il faut prévoir normalement dans un atelier arithmétique :
> - des organes relatifs aux nombres […]
> - des organes relatifs aux instructions […] (Kuntzmann 1957, p. 11, fascicule 1)

Les constituants de machine sont distingués selon les objets sur lesquels ils travaillent : nombres ou instructions.

Machines et mémoires

Kuntzmann fournit un schéma pour décrire l'Architecture d'une machine générique en relation avec son fonctionnement :



> La circulation des nombres et des instructions est en traits pleins ; celles des commandements et des actions qui enchaînent ces instructions sont en pointillés. (Kuntzmann 1957, p. 12, fascicule 1)

Dans ce schéma, il distingue deux types de mémoires celle des nombres et celle des instructions. Mais Kuntzmann précise plus loin la possibilité d'une seule mémoire dite « banalisée » qui gagne alors en capacité de stockage :

> Un dernier pas consiste à avoir, au lieu de deux mémoires ayant des caractéristiques technologiques identiques mais distinctes, une seule mémoire contenant à la fois les nombres et les instructions. On dit alors que la mémoire est banalisée. Ceci comporte entre autres avantages que : la capacité de mémoire est mieux utilisée puisque l'on pourra, suivant les besoin, loger beaucoup de nombres ou beaucoup d'instructions […] Avec mémoire banalisée : la plupart de grosses machines récentes américaine ou européenne entrent dans cette catégorie. (Kuntzmann 1957, p. 26, fascicule 1)

La classification des ateliers est reprise en relation avec le type de mémoires. Différentes caractéristiques des mémoires sont listées (cf. fascicule 1, p. 20) :

- Nature : connexion ou bouton ; cartes perforées ou ruban ; électrique ou magnétique ;
- Effaçabilité : une même position peut, au cours du calcul, être utilisée plusieurs fois pour des usages différents ; ou ineffaçabilité : chaque position ne peut donc être utilisée qu'une fois au cours d'un calcul ;
- Mode d'accès : direct ; défilement (cyclique, irréversible, réversible) ;
- Structure : tambour magnétique, électrique ;
- Homogénéité ou non de la mémoire;

Par exemple, la mémoire « carte perforée » est classée comme ineffaçable, à défilement irréversible (devenu cyclique si l'on réalimente le paquet).
Pour Kuntzmann, la caractéristique d'effaçabilité de la mémoire constitue un point fondamental pour l'exécution d'un programme :

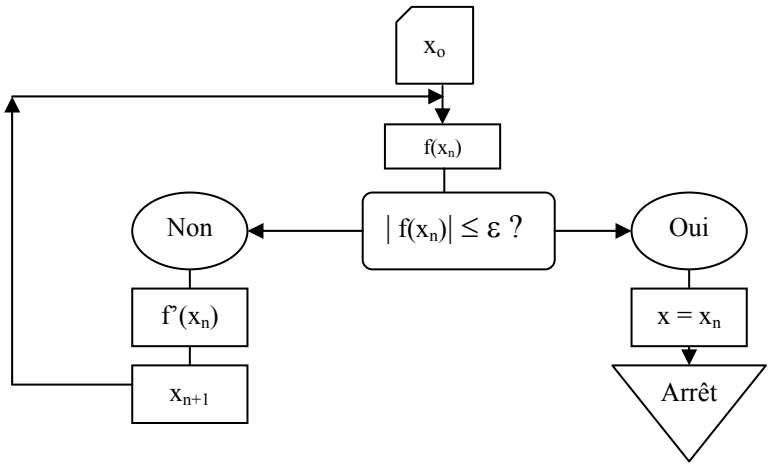
> Nous avons déjà dit que l'effaçabilité de la mémoire était une caractéristique essentielle de la catégorie la plus évoluée des machines. Cela entraîne la possibilité très intéressante de faire modifier, par le programme lui-même, certaines instructions en cours de calcul. (Kuntzmann 1957, p. 46, fascicule 1)

La notion de programme enregistré qui préside au concept de l'architecture de Von Neumann (voir partie C) n'est pas nominalement présente dans ce cours mais est pleinement explicitée dans cette citation au travers de la notion d'effaçabilité.

Effaçabilité d'une mémoire, notion de boucle

Le programme, moyen de communication entre l'homme et la machine, enregistré dans la mémoire, se substitue donc à l'intervention de l'homme sur la machine au cours d'un calcul.
La notion d'organigramme *déploy*é est un moyen didactique utilisé par Kuntzmann pour introduire la notion de boucle, visualisée par un cycle dans un organigramme (non déployé) : une boucle sert à « *éliminer des répétitions* ».

> Mais un tel organigramme déployé, non seulement contient un très grand nombre de répétitions, mais est même impossible à écrire explicitement à moins de connaître une borne supérieure du nombre d'itérations nécessaires. […] La disposition à adopter, pour éviter les répétitions est la suivante :



> On obtient ainsi ce que l'on nomme une boucle. Les boucles sont extrêmement fréquentes dans les programmes. (Kuntzmann 1957, p. 7, fascicule 2 – à propos du calcul d'une racine d'une équation algébrique par la méthode de Newton)

La notion de variable n'est jamais explicitée même au moment de la définition de la notion de boucle, mais elle est bien sûr implicitement présente dans l'écriture d'une boucle, tout spécialement à travers l'importance accordée par Kuntzmann à la caractéristique « effaçabilité » d'une mémoire. Cette dernière caractéristique conditionne aussi la possibilité de l'écriture d'une boucle :

> […] on peut le remplacer par un autre, beaucoup moins encombrant, en faisant une boucle, à condition que la mémoire soit effaçable. (Kuntzmann 1957, p. 7, fascicule 2)

L'écriture d'une boucle est ainsi liée intrinsèquement à la machine au travers de l'effaçabilité de sa mémoire.

> Autant que possible, on opèrera de la manière suivante :
> Les instructions de la boucle seront placées en séquence, la dernière étant le test. Ce test sera posé de telle manière qu'en répondant « oui » on continue dans la boucle ; […]
> Dans une machine utilisant des tests et non des comparaisons on aura souvent intérêt à réaliser la sortie de la boucle en constatant la formation d'un zéro. (Kuntzmann 1957, p. 10, fascicule 2)

Si Kuntzmann mentionne plusieurs machines de l'époque : GAMMA avec P.P.C, I.A.S, FERRANTI, UNIVAC, C.P.C MARK I, etc., il s'attache plus spécialement à la machine EDSAC (Electronic Delay Storage Automatic Computer), *machine non disponible à Grenoble.*
Cette machine, conçue au début de l'année 1947 par Wilkes M.V et mise en service en 1949 à l'Université de Cambridge, est le premier[1] ordinateur numérique, totalement *électronique* basé sur l'architecture de Von Neumann. La programmation en EDSAC avec un premier type de boucle, « go to » avec un test à zéro, est réalisée techniquement par Wheeler, informaticien anglais :

> As the first programmer for EDSAC, Wheeler invented ways of working which have now become standard. He realised that lines of program code could often be reused, and created the subroutine and the idea of keeping frequently-needed subroutines in a separate library which could be called on as necessary. He also developed the "Wheeler Jump" to allow a program to pass control to a subroutine, the precursor of the "go to" statement known to everyone who has ever written a program in Basic. (cf. http://.thocp.biographies/wheeler_david.htm)

I.3. Résumé

La notion de machine est au cœur de ce tout premier enseignement de programmation. Pour l'introduction à la programmation Kuntzmann choisit la machine EDSAC, la plus puissante de l'époque et qui permet l'écriture des boucles dans le langage le plus « compréhensible » parmi les langages des machines de l'époque. Nous pouvons dire que ce choix d'enseignement se caractérise par une machine idéale (c'est-à-dire non présente) et un langage idéal.
Les objets informatiques fondamentaux présents sont instructions, boucles, programmes, organigrammes et architecture de la machine. La notion de langage est mentionnée d'une manière intuitive, sans la distinction « langage machine » ou « langage assembleur ». Ce cours met en avant la relation étroite existant entre les caractéristiques d'une machine, en particulier celles de la mémoire, et la manière dans laquelle les programmes sont écrits. Le rôle de la mémoire est mis en valeur dans la classification des machines sans être attaché explicitement à la notion de variable, notion présente seulement implicitement dans les programmes avec boucles et dans la notion d'effaçabilité d'une mémoire. De ce point de vue chez Kuntzmann, une variable informatique est un emplacement dans une mémoire effaçable.
Le rôle des organigrammes est central dans ce traité pour la conception, la construction et l'évaluation des programmes et pour la conceptualisation de la notion de boucle. C'est en écrivant un organigramme avec boucle que la notion d'affectation de variable est présente. Nous y reviendrons dans le chapitre B3.

[1]Les cinq premiers ordinateurs sont l'EDVAC, la machine IAS, le BINAC, l'EDSAC et le MARK I à Manchester. L'ENIAC est l'un des premiers gros calculateurs électroniques, et non un ordinateur (cf. Breton 1990).

II. Knuth (1968) « Fundamental Algorithms »

Knuth publie en 1968, le premier volume « Fundamental Algorithms » (soit « Des algorithmes fondamentaux ») de son ouvrage, devenu référence dans la communauté d'informaticiens, « The art of Computer Programming[1] » (soit « L'art de la programmation d'ordinateur »). Le projet de Knuth est de publier 7 volumes : le deuxième est publié en 1969, le troisième en 1973 et une partie du quatrième volume en 2005. Knuth donne ses intentions dans la préface du premier volume :

> This set of books is intended for people who will more than just casually interested in computers, yet it is by no means only for the computer specialist. Indeed, one of the main goals has been to make these programming techniques more accessible to the many people working in other fields who can make fruitful use of computers, yet who cannot afford the time to locate all of the necessary information which is buried in the technical journals. (Knuth 1968, p. vi)

Entrons directement dans le premier volume « Fundamental algorithms ». Ce volume comprend deux grands chapitres découpés eux-mêmes des sous-chapitres comme suit :

> Chapitre 1-Concepts de base : 1.1. Algorithmes, 1.2. Introduction mathématique, 1.3. MIX, 1.4. Quelques techniques fondamentales en programmation ; Chapitre 2-Structure de données : 2.1. Introduction, 2.2. Listes linaires, 2.3. Arbres, 2.4. Structures multiconnectées, 2.5. Allocation dynamique de mémoire.

Le rôle attribué aux mathématiques et la relation intime de celles-ci avec l'informatique sont visibles tout le long de ce livre. L'auteur n'hésite pas à comparer la construction d'un programme à partir d'un ensemble d'instructions fondamentales avec celle de la construction d'une preuve mathématique à partir d'un ensemble d'axiomes :

> I wish to show that the connection between computer and mathematics is far deeper and more intimate than these traditional relationships would imply. The construction of a computer program from a set of basic instructions is very similar to the construction of a mathematical proof from a set of axioms. (Knuth 1968, p. x)

Plusieurs savoirs mathématiques sont présents tout au long du livre, en particulier au sein d'un sous-chapitre intitulé « 1.2. Introduction mathématique » :

> Induction ; nombres, puissance et logarithme ; somme et produit ; fonctions entières et théorie élémentaire des nombres ; permutation et factorielle ; coefficients binomiaux ; nombres harmoniques ; nombres de Fibonacci ; fonctions génératrices ; analyse d'un algorithme; représentation asymptotique. (traduit par nous)

Mais l'objectif de l'ouvrage est d'être en même temps un enseignement d'algorithmique et de programmation. D'algorithmique, car l'auteur aborde l'étude des algorithmes, des structures de données et l'analyse d'algorithmes. De programmation car l'auteur aborde les techniques de programmation.

II.1. Description des algorithmes et langage

Dès les années 70, plusieurs langages évolués et plusieurs marques d'ordinateur apparaissent, mais ces langages sont en cours de développement dans les diverses applications, et ne sont pas encore disponibles sur tous les ordinateurs.
Knuth va donc concevoir un langage orienté vers une machine (machine-oriented language) qu'il nomme Mixal. Il justifie longuement son choix, qu'il qualifie de « *the hardest decision which I had to make while preparing these books* ».

[1] À la fin de 1999, ces livres ont été appelés parmi les douze meilleures monographies de la physique-science du siècle par American Scientist, avec, entre autres : Einstein sur la relativité, Mandelbrot sur des fractals, Von Neumann et Morgenstern sur la théorie des jeux rectangulaires. (cf. Wikipédia 2005)

• Tout d'abord ce langage, indépendant des langages existants, fait l'économie de l'enseignement de connaissances préalables d'un langage particulier de programmation, permet une meilleure résolution des problèmes posés dans son livre et de rendre compte de la réalité physique de la machine.

> a) Algebraic languages are more suited to numerical problems than to the non numerical problems considered here; although programming languages are gradually improving, today's languages are not yet appropriate for topics such as co-routines, input-output buffering, generating random numbers, multiple-precision arithmetic, and many problems involving packed data, combinatorial searching, and recursion, which appear throughout.
> b) A programmer is greatly influenced by the language in which he writes his programs; there is an overwhelming tendency to prefer constructions which are simplest in that language, rather than those which are better for the machine. By writing in a machine-oriented language, the programmer will tend to use a much more suitable method; it is much closer to reality.
> c) The programs we require are, with a few exceptions, all rather short, so with a suitable computer the will be no trouble understanding the programs.
> d) A person who is more than casually interested in computers should be well schooled in machine language, since it is a fundamental part of a computer.
> e) Some machine language would be necessary anyway as output of the software programs described in chapters 1, 9, 10 and 12. (Knuth 1968, p. x)

• Le souci pédagogique fait aussi partie de ses arguments pour le choix de son langage orienté machine : assez puissant pour programmer brièvement de nombreux algorithmes et suffisamment simple pour être aisément appris :

> The language of MIX has been designed to be powerful enough to allow brief programs to be written for most algorithms, yet simple enough so that its operation are easily learned. (Knuth 1968, p. 120)

Contrairement à Kuntzmann qui ne définit pas la notion d'affectation de variable lors de l'introduction d'un algorithme itératif, l'attention pédagogique conduit Knuth à accorder une place importante à l'explicitation de la différence entre l'affectation d'une variable et l'égalité entre deux valeurs (ou variables) en mathématique. Il propose et justifie l'usage du symbole « ← » pour marquer cette différence. L'usage du signe « = » dans un programme est lié à une condition (mathématique) que l'on teste, le signe « ← » à une action qui peut être réalisée.

> The "←'' arrow […] is the all-important replacement operation (sometimes called assignment or substitution); "m ← n'' means the value of variable m is to be replaced by the current value of variable n. […] An arrow is used to distinguish the replacement operation from the equality relation: We will not say, ''Set m = n'' but we will perhaps ask, "Does m = n?'' The ''=" sign denotes a condition which can be tested, the "←" sign denotes an action which can be performed. The operation of increasing n by one id denoted by "n ← n + 1" (read "n is replaced by n + 1"); in general, "variable ← formula" means the formula is to be computed using the present values of any variables appearing within it, and the result replaces the previous value of the variable at the left of the arrow. (Knuth 1968, p. 3)

• Il affirme aussi son intention d'associer au langage Mixal d'autres outils bien connus de description d'un programme, à savoir l'organigramme et la description algorithmique étape par étape :

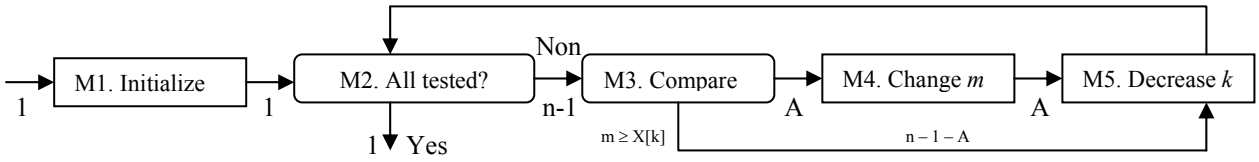
> The advantages of flowcharts and an informal step-by-step description of an algorithm are well known; (Knuth 1968, p. xi)

Les trois façons de décrire un algorithme (organigramme, description étape par étape, programme en langage Mixal) sont indissociables. L'exemple suivant éclaire ce propos et donne une idée plus précise de ce qu'est un programme en langage Mixal. Il s'agit du problème « déterminer le maximum et sa position dans une liste donnée de n éléments ». L'algorithme de résolution de ce problème est qualifié par Knuth de "typical algorithm".

> Algorithm **M** (*Find the maximum*). Given n elements X[1], X[2], …, X[n], we will find m and j such that $m = X[j] = \max_{1 \le k \le n} X[k]$ for which j is as large as possible.
> M1. [Initialize.] Set $j \leftarrow n$, $k \leftarrow n - 1$, $m \leftarrow X[n]$.

M2. [All tested?] If $k = 0$, the algorithm terminates.
M3. [Compare.] If X[k] ≤ m, goto M5.
M4. [Change m.] Set $j \leftarrow k$, $m \leftarrow$ X[k]. (Now m is the current maximum)
M5. [Decrease k.] Decrease k by *one*, return to M2. (Knuth 1968, p. 95)

Comme pour tous les algorithmes présents dans ce livre, un organigramme le succède (p. 95) :



Rappelons qu'à l'époque les organigrammes outillent la conception des programmes. La construction de tels organigrammes suit des règles strictes, dont certaines sont déjà présentes dans le cours de Kuntzmann. Ces règles sont développées dans des revues scientifiques en Informatique comme "Computer-Flow Draw charts" de Knuth (1963 in *ACM*, N°6) ou "Flowcharting techniques" (in revue d'*IBM*, 1969).
Dans son ouvrage, Knuth présente la règle Kirchhoff « dans un nœud il y autant d'entrées que de sorties » à l'occasion de cet organigramme :

> Labels on the arrows indicate the number of times each path is taken. Note that ''Kirchhoff's law'' must be satisfied, i.e., the amount of flow into each node must equal the amount of flow going out. (Knuth 1968, p. 95)

La boucle "Goto" qui est utilisée dans cet exemple est la seule présente dans le traité.

• Knuth justifie enfin le langage Mixal en affirmant la simplicité de la traduction d'un programme en Mixal en d'autres langages machines:

> Mixal has been designed so that it is a relatively simple matter to translate any Mixal programme into numeric machine language; (Knuth 1968, p. 141)

Pour décrire un algorithme, Knuth choisit ainsi un langage de programmation Mixal orienté machine, intrinsèquement lié à une description algorithmique étape par étape et à des organigrammes (comme Kuntzmann). Mixal n'étant pas un langage structuré, ces deux dernières façons de décrire un algorithme contribuent à la compréhension du programme en précisant le contenu de chaque instruction (description algorithmique étape par étape) et à une vision globale de la structure du problème derrière le programme et l'algorithme (organigramme) :

> Each step on an algorithm (e.g., step E1 above) begins with a phrase in brackets with sums up briefly as possible the principal content of that step. This phrase also usullay appears in accompanying flow chart […] so the reader will be able to picture the algorithm more readily. (Knuth 1968, p. 2)

Mais le choix du langage orienté machine étant affirmé et argumenté (longuement), quelle machine oriente ce langage ?

II.2. Machine et langage

À l'époque de la publication du premier ouvrage de Knuth, les langages évolués comme Fortran ou Algol étaient implémentés dans certaines machines. Par exemple, le langage Fortran dans les machines d'IBM corporation (Fortran I pour IBM 704, Fortran II pour IBM 7090/7094) (cf. IBM 1961) ou le langage Algol60 pour la machine B5000 de Burroughs Corporation[1]. La question de choisir la machine orientant le langage Mixal est donc posée.
Knuth introduit dans son ouvrage une machine *fictive MIX* . Il justifie son choix ainsi :

[1] C'est la première machine ordinateur dont la mémoire est organisée en pile permettant la récursivité en langage Algol. (cf. Wikipédia 2005)

> I could have chosen the language of a particular machine X, but then those people who do not process machine X would think this book is only for X-people. Furthermore, machine X probably has a lot of idiosyncrasies which are completely irrelevant to the material in this book yet which must be explained; and in tow years the manufacturer of machine X will no longer be of interest to anyone. […] To avoid this dilemma, I have attempted to design an "ideal" computer called "MIX" with very simple rules of operation (requiring, say, only an hour to learn), and which is also very much like nearly every computer now in existence. Thus MIX programs can be readily adapted to most actual machines, or simulates on most machines. (Knuth 1968, p. xi)

Knuth conçoit donc une machine idéale :

- universelle dans le sens où elle ressemble à toutes les machines existantes de l'époque ;
- permettant facilement l'adaptation des programmes écrits pour cette machine à d'autres machines.

Cette machine devra respecter des règles simples de fonctionnement que nous allons présenter en décrivant la structure de MIX.

La machine MIX comprend 3 parties essentielles : une mémoire, des registres, des dispositifs d'entrée-sortie. Elles sont décrites comme suit :

> The A-register has many uses, especially for arithmetic and operating on data. The X-register in an extension on the ''right-hand side'' of rA, and it is used in connection whit rA to hold ten bytes of a product or dividend, or it can be used to hold information shifted to the right out of rA. The index registers rI1, rI2, rI3, rI4, rI5 and rI6 are used primarily for counting and for referencing variable memory addresses. The J-register always holds the address of the instructions following the preceding ''JUMP'' instruction, and it is primarily used in connection with subroutines. Besides these registers, MIX contains
> - an *overflow toggle* (a single bit which is either ''on'' or ''off'),
> - a *comparison indicator* (which has three values: less, equal, or greater),
> *memory* (4000 words of storage, each word with five bytes plus sign)
> - and *input-output devices* (card, tape, etc.). (Knuth 1968, p. 122)

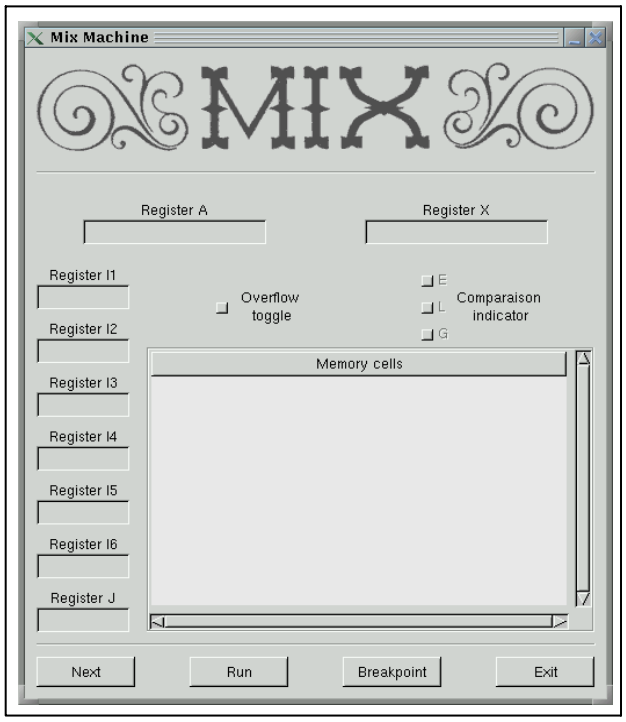


Figure 1. Machine Mix (image disponible sur http://valvassori.free.fr/p_unix.php3)

Knuth décrit donc la structure de la machine MIX comme un ensemble de neuf registres et d'une seule mémoire. La capacité de la mémoire joue un rôle important dans la conception de la machine :

> An algorithm in MIX should work properly regardless of how big a byte is. (Knuth 1968, p. 121)

La connaissance de l'architecture de MIX permet de comprendre les interrelations entre langage Mixal et le fonctionnement de MIX.

Revenons au programme associé à l'algorithme de résolution du problème « déterminer l'élément maximum d'un ensemble d'éléments » (p. 141):

> Program **M** (Find the maximum). Register assignments : rA ≡ m, rI1 ≡ n, rI2 ≡ j, rI3 ≡ k, X[i] ≡ cell(X + i).

Assembled instructions

3000 :	+	30	09	0	2	32
3001 :	+		0	1	2	51
3002 :	+	30	05	0	0	39
3003 :	+	10	00	3	5	56
3004 :	+	30	07	0	7	39
3005 :	+		0	3	2	50
3006 :	+	10	00	3	5	08
3007 :	+		1	0	1	51
3008 :	+	30	03	0	2	43
3009 :	+	30	09	0	0	39

Line no	LOC	OP	ADDRESS	Times	Remarks
01	X	EQU	1000		
02		ORIG	3000		
03	MAXIMUM	STJ	EXIT	1	Subroutin linkage
04	INIT	ENT3	0,1	1	M1. Initialize. k←n-1
05		JMP	Changem	1	j←n, m←X[n],k←n-1
06	LOOP	CMPA	X,3	n-1	M3. Compare.
07		JGE	*+3	n – 1	
08	CHANGEM	ENT2	0,3	A+1	M4. Change. j←k
09		LDA	X,3	A+1	m←X[k]
10		DEC3	1	N	M5. Decrease k.
11		J3P	LOOP	N	M2. All tested?
12	EXIT	JMP	*	1	Return to main program.▪

Knuth associe une double écriture du programme : en langage Mixal et en « *Assembled instructions* », désigné par lui comme « numeric machine *language ».* À tout moment de l'exécution d'un programme, il est possible de connaître l'état « physique » de la machine, c'est-à-dire l'état de la mémoire et des registres.
Chaque instruction en langage Mixal correspond donc à une instruction numérique de la machine MIX. Illustrons sur un exemple le changement opéré dans la mémoire par une instruction numérique en langage machine
Soit l'opération « charger le A-registre avec le champ (*field*) (1 :3) ». Supposons que la case indicée 2000 dans la mémoire contienne :

–	8	0	3	5	4

Voici le changement du contenu du registre A opéré suivant les instructions numériques :

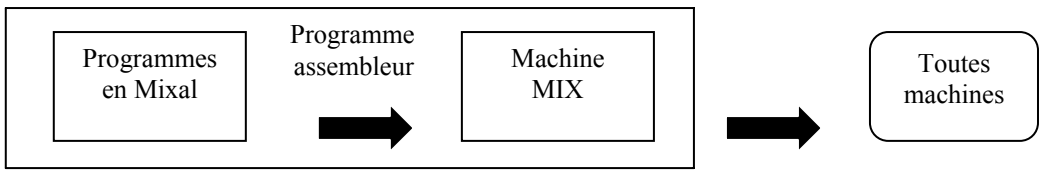
Représentation conventionnelle	Instruction numérique						Le contenu du registre A (rA) après l'instruction					
	±	A	A	I	F	C						
LDA 2000(1 : 5)	+	20	00	0	13	8	+	8	0	3	5	4
LDA 2000(0 : 3)	+	20	00	0	3	8	–	0	0	8	0	3

Figure 2. Changement opéré dans la mémoire de la machine MIX

Ainsi Knuth veut que l'élève-lecteur accède à l'état physique la machine ordinateur, pour comprendre comment l'ordinateur exécute les programmes afin de pouvoir comparer l'efficacité et le coût des différents programmes. Pour Knuth, une telle compréhension du processus d'exécution des programmes à l'aide de cette « simulation » est non seulement utile mais nécessaire à la conception et à la compréhension d'un programme informatique :

> The reader will find it instructive to study as many of the MIX programs given in this book as possible - it is exceedingly important to acquire skill in reading other people's computer programs, yet such training has been sadly neglected in too many computer courses and it has led to some horribly inefficient uses of computing machinery. (Knuth 168, p. 170)

La machine MIX étant conçue par Knuth comme représentante de toutes les machines existantes, le langage machine numérique (numeric machine language) de cette machine fonctionne sur toutes autres machines. Le schéma suivant illustre ce propos :



II.3. Résumé

Le corps des savoirs présents dans le traité de Knuth est organisé autour d'un couple « universel » : la machine idéale MIX et le langage Mixal orienté vers MIX.
L'architecture de cette machine, décrite sous forme d'un ensemble de neuf registres et d'une unique mémoire, est aussi un objet de savoir fondamental. La double écriture de tout programme - langage Mixal et « numeric machine language » - permet à l'élève-lecteur une simulation de l'exécution de ces programmes à deux niveaux de la machine et donne accès au suivi de l'état « physique » des mémoires ainsi que des registres. Pour Knuth, la conception d'un programme repose sur le processus d'explicitation des états de la machine (registres et mémoire).
Knuth développe un discours explicatif sur la notion de variable, au travers de la notion d'affectation ; en particulier il prend soin de la distinguer son usage de celle de variable mathématique.
Comme chez Kuntzmann, la boucle "Goto" est la seule présente. Nous faisons l'hypothèse que la raison en est que le langage est structuré par le fonctionnement de la machine et non pas par les trois structures de contrôles : séquentialité, branchement et itération (comme c'est le cas dans l'enseignement actuel). La description de l'algorithme par un organigramme avec son système de règles d'écriture établit naturellement cette boucle.

III. Horowitz et Sahni (1978) « Fondamentals of Computer Algorithms »

L'ouvrage de Horowitz et Sahni, intitulé « Fondamentals of Computer Algorithms » (soit « Fondements des Algorithmes Informatiques ») a pour objectif d'organiser les connaissances du moment sur les algorithmes pour permettre leur conception et leur analyse :

> Organiser ce que l'on connaît sur les algorithmes à l'heure actuelle dans une manière cohérente, de telle façon que étudiants et praticiens puissent concevoir et analyser de nouveaux algorithmes eux-mêmes. (Horowitz et Sahni 1978, p. vii) (Traduit par nous)

On peut voir dans cet objectif une définition de ce qu'est l'algorithmique pour Horowitz et Sahni.
Leur ouvrage est composé de 12 chapitres qui se déclinent comme suit :

> 1. Introduction ; 2. Structure de données élémentaires ; 3. Diviser et conquérir ; 4. La méthode gourmande ; 5. Programmation dynamique ; 6. Recherche de base et techniques transversales ; 7. Retour en arrière ; 8. Branche et borne ; 9. Simplification et transformation algébrique ; 10. Théorie de la borne inférieure ; 11. NP-durs et NP-complets problèmes ; 12. Approximation algorithmique pour NP-durs problème. (Traduit par nous)

Ce livre commence donc par présenter des algorithmes considérés comme fondamentaux ainsi que des structures de données pour aborder ensuite la conception et l'analyse des algorithmes.

III.1. Langage

Pour conduire cet enseignement, les auteurs inventent un langage spécifique de description des algorithmes, le langage SPARKS. Ils justifient leur choix ainsi :

> Premièrement, nous souhaitons pouvoir écrire nos algorithmes sans devoir penser sans cesse à des caractéristiques propres au langage choisi. Deuxièmement, chaque langage a ses disciples et ses détracteurs. Nous pouvons avoir des lecteurs qui ne connaissent pas ce langage ou, plus particulièrement, qui n'aiment pas ce langage. De plus, il n'est pas nécessaire d'écrire un algorithme dans un langage de programmation dont le compilateur existe déjà. Plus le langage est suffisamment proche de plusieurs langages existants déjà mentionnés (Algol, Fortran, Lips, Pascal et PL/I etc.), plus une traduction à la main est relativement facile à accomplir. Ceci nous encourage à développer un langage simple qui est taillé pour la description des algorithmes qui vont être discutés. De cette manière, nous ne devons pas aborder des aspects en langage de programmation qui ne sont jamais utilisés ici. Nous l'appelons

> SPARKS. Dans sa forme, ce langage est proche aux langages Pascal et Algol 60. (Horowitz et Sahni 1987, p. 5) (Traduit par nous)

Ce langage est donc choisi pour se libérer des particularités des langages de programmation existants et de leur apprentissage préalable, tout en étant suffisamment proche des langages existants pour faciliter sa traduction ultérieure. Ce langage est en particulier proche du langage Pascal.
Ce choix étant affirmé et argumenté, les programmes de l'ouvrage ne sont exprimés qu'en ce langage. Nous donnons ci-après un exemple d'un tel programme (p. 14) :

```
y ← 0
while y ≤ n do
read(x)
y← y + x
repeat
```

Quatre types de boucles sont présentes dans l'ouvrage, ce que les auteurs justifient par la nécessité de faciliter la tâche de programmation :

> Il est bien connu que tous les langages « propres » peuvent être écrits en utilisant seulement l'affectation, la condition et l'instruction « while ». Bien que ceci soit très intéressant du point de vue théorique, nous ne le retiendrons pas comme la méthode dans laquelle nous programmons. Au contraire, plus les langages sont expressifs, plus nous pouvons accomplir la tâche de programmation facilement. (Horowitz et Sahni 1987, p. 8) (Traduit par nous)

Ils présentent alors, en plus de la boucle : « *While…do* », trois autres boucles : « *Loop …until cond repeat* », « *For vble ← start to finish by increment do … repeat* », et « *go to label* ». Chaque boucle est accompagnée d'un organigramme.
Par contre, *la notion de variable n'est pas explicitée*, les organigrammes peu présents et la description algorithmique étape par étape complètement absente.
Le choix de présenter toutes ces boucles ainsi que leur conception sur ce qu'est un bon langage « *un langage est bon s'il nous permet de décrire l'abstraction du monde réel d'une manière naturelle* » (p. 614) permettent de penser que le critère est aussi la proximité du langage SPARKS avec le langage naturel.
Rappelons que dans les années 80, les travaux contemporains de Dijkstra, Hoare, Jacopini (1972) concernant la programmation structurée ont des conséquences sur l'effacement de la boucle « si … alors aller à » et de l'organigramme qui lui est naturellement associé.

> Le langage algorithmique défini ci-dessus est largement inspiré du langage des D-schémas introduit par Dijkstra, Hoare dans les années 1970 (Structured Programming, 1972, Academic-Press). Leur approche est justifiée par un article de Ashcroft et Manna (1971) qui établit que tout ordinogramme de conception séquentielle avec branchement aléatoire (l'ancien go to) peut être ramené à un langage utilisant l'itération tant que (while). Le langage des D-schémas a la propriété de s'exprimer comme une suite d'instructions sans aucun renvoi, ce qui rend inutile les ordinogrammes. (Cardon et Charras 1996, p. 85)

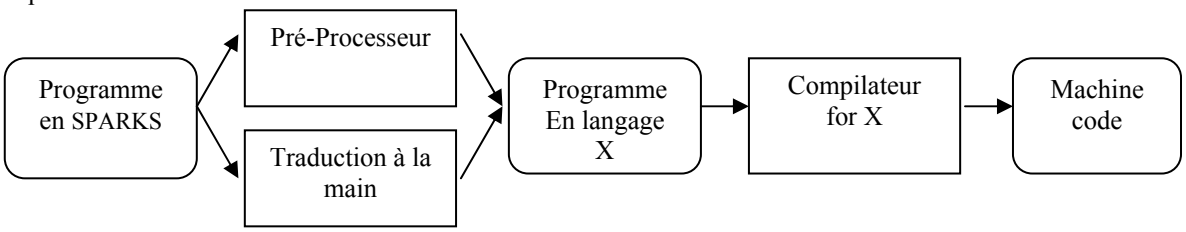
Ceci pourrait expliquer la rare présence des organigrammes dans l'ouvrage d'Horowitz et Sahni ainsi que la ressemblance du langage SPARKS avec le langage algorithmique structuré par les trois structures fondamentales que sont la séquentialité, le branchement et l'itération.

III.2. Machine et langage

Aucune machine spécifique n'est mentionnée dans cet ouvrage. Que disent les auteurs sur les relations entre le langage SPARKS et la notion de machine ?

D'abord les auteurs répondent à la question « sur quelle machine est exécutable un programme écrit en langage SPARKS ? » par : « sur n'importe quelle machine ».

La figure suivante montre comment un programme en langage SPARKS peut être exécuté sur n'importe quelle machine.



(Horowitz et Sahni 1987, p. 5) (Traduit par nous)

Le schéma montre que l'exécution d'un programme en langage SPARKS nécessite soit un pré-processeur, soit un traducteur humain, pour traduire ce programme en langage X. Pour Horowitz et Sahni, le pré-processeur traduit le langage SPARKS en langage Fortran pour les raisons suivantes :

1) Ce langage est souvent disponible sur des machines et son compilateur fonctionne souvent très bien ; 2) Ce langage permet un certain degré de portabilité que ne possèdent pas d'autres langages ; 3) Ce langage a une riche librairie de sous programmes et 4) Il y a une très forte familiarité à ce langage. (Horowitz et Sahni 1987, p. 615) (Traduit par nous)

Ce choix montre que l'apprentissage du langage SPARKS ne peut éviter la connaissance préalable d'un autre langage de programmation, ici Fortran.

III.3. Résumé

Le corps de savoirs présents dans le traité d'Horowitz et Sahni est donc organisé autour d'un langage algorithmique : le langage SPARKS.
Ce langage, dans lequel tous les programmes sont ou devront être exprimés, est visiblement influencé par le langage de programmation structurée Pascal.
La notion de variable reste implicite, mais la notion de boucle est un objet fortement présent et contribue pour Horowitz et Sahni à l'expressivité et la simplicité du langage SPARKS.
La notion de machine passe à l'arrière plan et disparaît des enjeux didactiques de ce manuel. Les auteurs invoquent un pré-compilateur permettant de traduire automatiquement un programme en langage SPARKS en un langage Fortran qui existe sur toutes les machines de l'époque. De ce point de vue, le langage SPARKS est universel.

IV. Conclusion de l'analyse des traités

Kuntzmann choisit pour son enseignement de la programmation, d'écrire et de faire écrire les programmes en langage assembleur d'une machine concrète et réelle EDSAC *évoquée*, puisque absente de l'environnement des étudiants. Rappelons qu'à cette époque, l'informatique n'était qu'au début de son histoire, l'ordinateur coûtait encore très cher, quelques rares universités possédaient un unique ordinateur.
Knuth construit une machine idéale, nommée MIX, associée à son langage assembleur Mixal. Le couple (machine MIX, langage Mixal) est conçu par Knuth comme universel :

- d'une part l'architecture de la machine est celle des machines existantes,
- d'autre part, un programme-machine est présent et permet de traduire tous les programmes écrits en Mixal en langage machine numérique.

Quant à Horowitz et Sahni, ils conçoivent un langage de programmation, SPARKS pour décrire tous les algorithmes. Ce langage évolué, proche du langage Pascal, est considéré comme universel car traduisible (par un pré-compilateur) en un programme en langage Fortran, disponible sur tous les ordinateurs de l'époque.

L'analyse de ces trois traités nous permet donc de repérer différentes organisations possibles d'un enseignement d'algorithmique et de programmation, vis-à-vis du rôle et du statut de la machine. Machine et langage plus ou moins dépendant de la machine sont présents dans les deux traités de Kuntzmann et Knuth, alors que la machine disparaît au profit du langage structuré chez Horowitz et Sahni.
Nous schématisons ces différentes organisations dans la figure 3 ci-après :

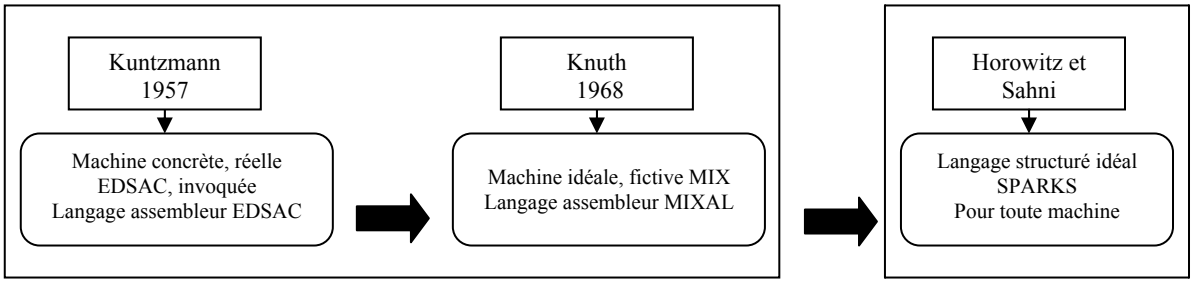


Figure 3. Les différentes organisations d'enseignement présentes dans trois traités

Dans le traité de Kuntzmann, deux manières de décrire un programme sont présentes avec des statuts différents : l'organigramme qui joue un rôle non seulement pour décrire, pour construire mais aussi pour évaluer un programme et le programme en langage assembleur EDSAC qui est le résultat de la tâche de programmation. Knuth ajoute une troisième description aux deux précédentes, la description algorithmique étape par étape. Dans ces deux traités, la notion de boucle est présente en tant qu'enjeu didactique, mais réduite à la seule forme « si … alors aller à ».
Par contre, Horowitz et Sahni en se centrant sur la description des algorithmes en langage SPARKS enrichissent les possibilités de programmation par l'introduction de 4 types de boucle.
L'analyse des traités nous permet donc de repérer deux stratégies possibles d'enseignement de l'algorithmique et de la programmation. Le terme de stratégie désigne pour nous une façon d'organiser le corps des savoirs dans un enseignement d'algorithmique et de programmation.

Première stratégie : présence d'une machine de référence, réelle ou fictive

Ces machines fictives peuvent être virtuelles (c'est-à-dire non construite mais particulière comme celle de Babagge, 1842), ou idéale (c'est-à-dire non construite mais représentante d'une classe de machines réelles comme la machine MIX de Knuth, 1968). L'enseignement de l'algorithmique et de la programmation est alors lié à la compréhension de l'architecture de la machine et à la conception d'un langage orienté vers cette machine.

Deuxième stratégie : absence de machine de référence

Dans cette stratégie, l'enseignement de l'algorithmique passe par celui d'un langage de programmation représentatif d'une classe de langages existants (comme le langage SPARKS de Horowitz et Sahni, 1978, langage évolué de style impératif). Cet enseignement suppose la connaissance préalable d'un langage de programmation.

Nous terminons ce chapitre par une brève étude sur l'état de l'enseignement d'informatique au début de l'université en France et au Viêt-nam, en tentant de répondre aux questions suivantes : quelle stratégie d'enseignement est actuellement prépondérante, la première ou la deuxième ? Quelles sont les raisons données au choix d'une stratégie ? Quels statuts et quelles places ont les objets fondamentaux « boucles » et variables » dans les enseignements au début de l'Université ?

V. Rapide tour d'horizon de l'enseignement de l'informatique au début de l'université

Nous avons entrepris une analyse des programmes et de quelques manuels concernant l'introduction de l'algorithmique et de la programmation au début de l'université en France et au Viêt-nam (pour les détails de cette analyse voir en Annexe B1). Nous nous contentons de donner ici quelques résultats de cette analyse :

Langage

En France, l'enseignement de l'algorithmique implique l'algorithmique et la programmation. Le langage algorithmique enseigné est à la fois proche d'un langage structuré de style impératif existant et représentatif d'une classe plus large de langages existants. Son enseignement exige des connaissances préalables d'un langage de programmation.
Au Viêt-nam, il n'y a pas l'enseignement d'algorithmique dans l'introduction de l'informatique. Certains auteurs assimilent des langages de programmation aux langages algorithmiques. À ce propos, au Viêt-nam, le seul style de programmation est le style impératif et le langage Pascal est le seul langage de programmation enseigné. En France le style de programmation dominant est aussi le style impératif mais plusieurs langages sont enseignés comme Pascal, C, Java etc. Les organigrammes sont présents dans plusieurs manuels au Viêt-nam alors qu'ils sont rarement présents dans les manuels utilisés en France.

Variable et boucle

Au Viêt-nam dans l'enseignement d'un langage de programmation, l'apparition des notions de variable (dans une affectation) et de boucle (dans des instructions itératives) arrive assez tard, même si le moment de cette introduction varie d'un livre à l'autre. Alors qu'en France, les notions de boucles et de variables sont introduites presque dès le début de l'enseignement, juste après l'introduction du langage de programmation. Voici par exemple une définition canonique des objets variable et boucle :

> Une variable sert à stocker une valeur. Une variable est caractérisée par son NOM, son TYPE, et sa valeur […] Une boucle est une séquence d'instructions à répéter. (Maysonnave 1996)

Les objets « variable » et « boucle » constituent des objets fondamentaux dans l'enseignement actuel de programmation du style impératif et ils restent présents tout le long de cet enseignement.
En France, les deux types invariants de boucle des langages algorithmiques enseignés sont : « tant que…faire » et « pour ». Certains manuels ajoutent un troisième type de boucle « répéter…jusqu'à ». La disparition annoncée de la boucle « si…alors aller à » est chose faite. Au Viêt-nam, à côté de ces trois types de boucles, ce type de boucle « si…alors aller à » reste encore présent mais son usage est accompagné de mise en garde de la part des auteurs.

Structures de données

En France, aux types de données élémentaires (nombres entiers, nombres réels, chaînes de caractère etc.) s'ajoute un autre type de données, les tableaux. D'autres types de données comme enregistrement, fichiers séquentiels etc. sont abordés mais leur présence varie d'un manuel à l'autre. Alors qu'au Viêt-nam tous ces types de données et d'autres comme ensemble et pointeur, sont présents dans l'enseignement universitaire.

Machine

La notion de machine accompagne très peu voire pas du tout cet enseignement que ce soit en France ou au Viêt-nam : les notions enseignées comme variable et boucle sont introduites en

l'absence de toute machine, autrement dit il n'y a pas de machine de référence à cet enseignement.

La stratégie 2, celle initiée par le traité d'Horowitz et Sahni, s'est imposée et on reconnaît dans les enseignements analysés les principales caractéristiques de cette stratégie. Or notre analyse de l'émergence d'un enseignement d'informatique montre une autre stratégie d'enseignement qui s'organise autour d'un couple machine de référence, (réelle ou fictive) et langage. Cette stratégie peut faire l'économie de l'apprentissage d'un langage particulier de programmation. Récemment, Knuth (2005) plaide pour un retour à cette stratégie d'enseignement qu'il a lui-même initié dans le premier volume de « The art of Computer Programming » :

> But the reasons for machine language that I gave in the preface to Volume 1, written in the early 1960s, remain valid today:
> - One of the principal goals of my books is to show how high-level constructions are actually implemented in machines, not simply to show how they are applied. […]
> - The programs needed in my books are generally so short that their main points can be grasped easily.
> - People who are more than casually interested in computers should have at least some idea of what the underlying hardware is like. Otherwise the programs they write will be pretty weird.
> - Machine language is necessary in any case, as output of many of the software programs I describe.
> - Expressing basic methods like algorithms for sorting and searching in machine language makes it possible to carry out meaningful studies of the effects of cache and RAM size and other hardware characteristics (memory speed, pipelining, multiple issue, look aside buffers, the size of cache blocks, etc.) when comparing different schemes.
> Moreover, if I did use a high-level language, what language should it be? In the 1960s I would probably have chosen Algol W; in the 1970s, I would then have had to rewrite my books using Pascal; in the 1980s, I would surely have changed everything to C; in the 1990s, I would have had to switch to C++ and then probably to Java. In the 2000s, yet another language will no doubt be de rigueur. I cannot afford the time to rewrite my books as languages go in and out of fashion; languages aren't the point of my books, the point is rather what you can do in your favourite language. My books focus on timeless truths. (Knuth 2005)

Nous reviendrons, au moment de la formulation de nos hypothèses de recherche sur ces deux stratégies.

Les deux stratégies repérées se distinguent par la place de la notion de machine et du langage de programmation dans l'enseignement de l'algorithmique et de la programmation.

Comment ces deux stratégies dépendent-elles de l'évolution historique des machines et des langages ?
Pour donner des éléments de réponse, nous faisons une lecture épistémologique des articles de spécialistes de l'histoire des machines et des langages : nous cherchons à repérer les problèmes qui ont provoqué ruptures et filiations dans la co-genèse de la notion de machine ordinateur et de la programmation des algorithmes (chapitre B2) puis dans la genèse des langages machines aux langages de programmation évolués (chapitre B3).

Chapitre B2
Machines et programmes

Introduction

Dans ce chapitre, nous nous intéressons à la genèse de la machine ordinateur que nous découpons en plusieurs étapes. La grille de notre analyse repose sur des questions suivantes : Quels problèmes mathématiques, dont on connaît une solution mathématique, sont associés aux différentes étapes de cette genèse jusqu'à la machine ordinateur actuelle ? Comment évoluent, d'une étape à l'autre, les rapports entre l'homme et la machine. Plus précisément, quelles sont les conditions spécifiques à chaque étape qui permettent à l'homme de dévoluer à la machine l'exécution de tâches mathématiques ?

I. Première étape : machines arithmétiques

Les premières machines automatiques utilisent des mécanismes d'horlogerie avec roues dentées et ergots, mécanismes appelés jacquemarts. Ce sont ces mécanismes qui actionnent des personnages pour exécuter des séquences de gestes identiques en fonction des heures de la journée. C'est dans ce même contexte technologique que sont inventées les machines arithmétiques.
La motivation première pour mécaniser des calculs arithmétiques est de chercher à se libérer du coût de ces calculs et de les rendre plus fiables.

> L'histoire des machines à calculer commence en Allemagne et en France au XVII siècle avec le reporteur. Ce mécanisme qui, pour la première fois, libère d'une opération de calcul mental répétitive et fastidieuse (le report des dizaines ou la retenue) va être à l'origine du développement de toutes les machines ultérieures. (Marguin 1993)

Considérons l'addition : 1962 + 59. Si l'on veut confier cette opération à une machine, il faut, dans l'arithmétique décimale, que la machine ait un moyen de garder les retenues résultant de l'addition des chiffres qui sont tous égaux à 1. La mécanisation des calculs va alors de pair avec le processus de mémorisation qui a été matérialisé dans ces machines par le reporteur.

En 1623, Schickard, astronome allemand, conçoit la première machine exécutant automatiquement les additions et les soustractions. Dans cette machine, un ergot entraîne la roue des dizaines.

> Cette additionneuse était constituée de roues dentées ayant chacune dix dents : soit à additionner 2 nombres d'un seul chiffre décimal, la roue tourne dent par dent pour totaliser les unités ; lors qu'elle arrive entre les dents marquées 9 et 0, elle fait tourner d'un cran la roue des dizaines. (Moreau 1987, p. 13)

Nous pouvons interpréter ce report mécanique comme une structure conditionnelle *matérialisée* : si la somme de 2 nombres est supérieure ou égale à 10, alors faire tourner d'une unité la roue des dizaines. Le seul modèle de la machine Schickard, construit de son vivant, ainsi que les documents la concernant disparaissent dans un incendie en 1924. Des lettres envoyées à l'astronaute Kepler dans lesquelles il décrit sa machine ne sont retrouvées en Russie qu'en 1958 grâce à quoi plusieurs répliques de sa machine sont construites (cf. Ligonnière 1987).
Cette invention n'a pas pu influer sur les machines ultérieures. Nous allons maintenant examiner trois machines qui, en filiation les unes avec les autres, font évoluer la machine

arithmétique sans toutefois introduire de véritables ruptures dans son architecture et dans ses relations avec l'opérateur : les machines de Pascal, Leibniz et la machine à différences finies de Babbage.

I.1. Le problème de la mécanisation des quatre opérations

I.1.1. La machine « additionneuse » de Pascal

En 1642, afin d'aider son père dans des calculs arithmétiques, Pascal imagine une machine arithmétique, la « Pascaline » qui peut exécuter automatiquement l'addition, la soustraction et aussi les conversions des monnaies de l'époque.

> Il cherchait à faire reposer les additions sur un simple jeu d'engrenages, mais souhaitait traiter également la soustraction. (Ligonnière 1987, p. 30)

Concernant le problème des retenues, il conçoit le « sautoir », organe spécial destiné à servir de liaison occasionnelle entre deux roues successives.

> Pendant plus de trois siècles, la Pascaline a été considérée comme la première machine à calculer de l'histoire. (Ligonnière 1987, p. 31)

La figure 1 représente certaines parties de la machine arithmétique : lucarnes, roues dentées, baguette mobile.

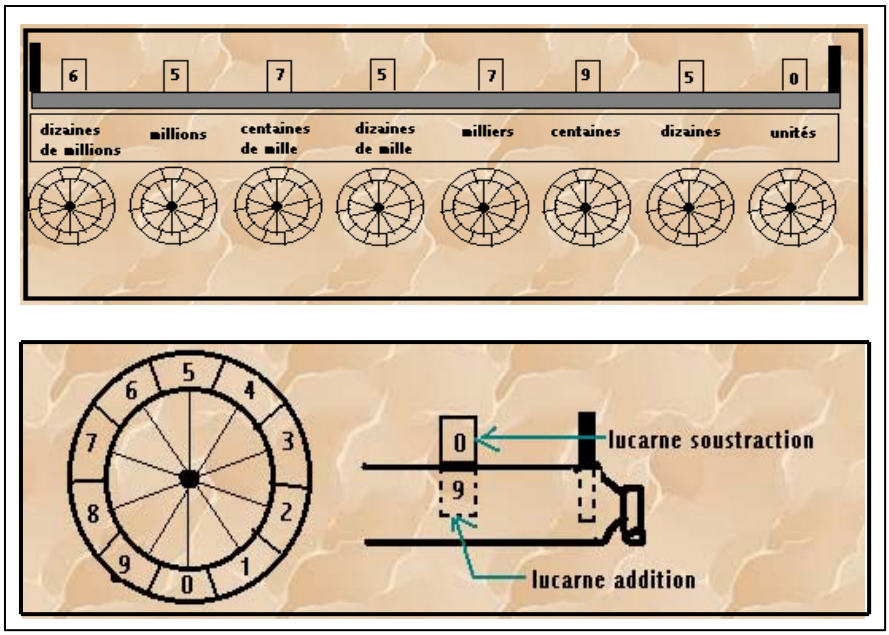


Figure 1. Une partie d'une Pascaline (image disponible sur http://histoiredechiffres.neuf.fr/)

La figure 1 montre des roues (dont le nombre varie de 5 à 10 selon le modèle de Pascaline, 20 modèles ayant été fabriqués, cf. CRDP Clermont-Ferrand 1981) sur le limbe desquelles sont gravés de manière régulière des chiffres de 0 à 9.

> Ces roues sont chacune situées au-dessous d'une lucarne ou apparaît un des chiffres. On place un poinçon entre les dents d'une roue, en fait d'un chiffre du limbe ; lorsque, à l'aide du poinçon, on fait tourner la roue dans le sens des aiguilles d'une montre, celle-ci entraîne un tambour à l'intérieur du boîtier, sur lequel sont inscrits les chiffres visibles à travers la lucarne. (Descotte 2003, p. 19)

Le résultat s'affiche aussi dans des lucarnes. Pour passer du mode addition au mode soustraction, il suffit de faire glisser la baguette (indiquée en grisé sur la figure) du haut vers le bas[1].

Décrivons sur un exemple comment la machine effectue l'addition 162 + 59.
La baguette est en position haute. Tous les chiffres des lucarnes sont mis à zéro. Les chiffres 2, 6 et 1 sont inscrits successivement et respectivement dans les lucarnes des unités, des dizaines et des centaines. Le premier nombre donné, 162, est ainsi enregistré dans la machine. Puis on fait de même pour le nombre 91. La machine distingue le deuxième nombre du premier grâce à l'ordre d'inscription sur la roue des unités. Suite à l'inscription du chiffre 9 sur la roue des unités, le nombre 171, premier résultat, apparaît dans les lucarnes, puis suite à l'inscription du chiffre 5 sur la roue des dizaines, le nombre 221, dernier résultat, le remplace.
La décomposition 162 + 9 = 171, puis 171 + 50 = 221 est intégrée dans la conception mécanique de la machine.
Nous formalisons la propriété mathématique sur laquelle s'appuie Pascal comme suit :

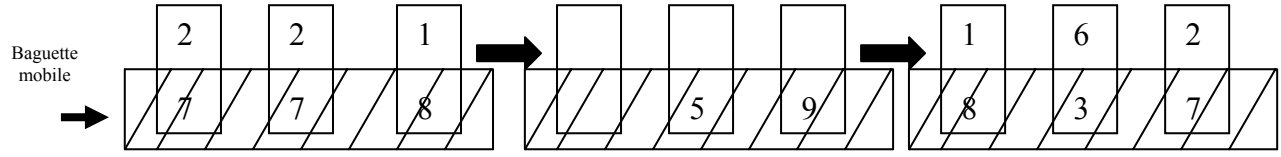
Pour tous deux nombres naturels $\overline{a_n a_{n-1}...a_1 a_0}$ et $\overline{b_m b_{m-1}...b_1 b_0}$, on a :

$$\overline{a_n a_{n-1}...a_1 a_0} + \overline{b_m b_{m-1}...b_1 b_0} = ((((\overline{a_n a_{n-1}...a_1 a_0} + b_0) + \overline{b_1 0}) + ...) + \overline{b_m 00...0})$$

Examinons maintenant la soustraction.
Pascal introduit dans sa machine une baguette mobile à deux positions (haute et basse) correspondant respectivement à l'addition et à la soustraction.
Les tambours intérieurs portent deux lignes de chiffres de 0 à 9, rangés dans l'ordre croissant en bas et décroissant en haut. La somme de deux chiffres, rangés l'un au dessous de l'autre, est donc égale à 9 comme le montre le dessin ci-dessous.

9 8 7 6 5 4 3 2 1 0
0 1 2 3 4 5 6 7 8 9

Soit à calculer 221 – 59.
La baguette en position basse. Tous les chiffres visibles dans les lucarnes sont mis à 9. L'opérateur calcule *mentalement* le complément de 9 de tous les chiffres composant le nombre 221. Ce sont respectivement 7, 7 et 8. Les chiffres 8, 7, 7 sont inscrits successivement et respectivement dans les lucarnes des unités, des dizaines et des centaines et *restent cachés* pas la baguette : les chiffres visibles sont 2, 2, 1. Puis on inscrit 9, 5 dans les lucarnes des unités et des dizaines. Le résultat intermédiaire 778 + 59 = 837 est calculé mécaniquement et *reste caché* par la baguette. Le résultat final 2, 6, 1 apparaît dans les lucarnes comme complément à 9 de 837. Le schéma suivant fait apparaître chronologiquement le jeu du visible et du non visible :



Ce processus s'appuie sur la propriété mathématique suivante :

[1] Dans la figure 1 la baguette est en position basse et la Pascaline est en mode soustraction.

Pour tous deux nombres naturels $\overline{a_n a_{n-1}...a_1 a_0}$ et $\overline{b_m b_{m-1}...b_1 b_0}$, on a :

$$\overline{a_n a_{n-1}...a_1 a_0} - \overline{b_m b_{m-1}...b_1 b_0} = \overline{99...99} - ((\overline{99...99} - \overline{a_n a_{n-1}...a_1 a_0}) + \overline{b_m b_{m-1}...b_1 b_0})$$

La baguette mobile et la disposition des chiffres en deux rangées sur chaque tambour a réalisé mécaniquement la différence : $\overline{99...99} - A$ avec A entier naturel.
Dans la soustraction, l'opérateur intervient davantage que dans l'addition : il doit faire glisser la baguette et trouver mentalement les compléments à 9 des chiffres composant le premier terme. Ce calcul mental peut induire des erreurs sur le processus de calcul.
Les opérations, division et multiplication, se ramenant respectivement à des soustractions ou à des additions, il n'y a besoin que de concevoir un dispositif pour mémoriser le nombre de soustractions ou d'additions effectuées. Pour cela, Pascal prévoit une série de rondelles sur la baguette pour garder une mémoire mécanique du nombre d'opérations :

> L'utilisateur va inscrire, à l'aide de ces rondelles, le nombre d'opérations qu'il exécute, c'est-à-dire le multiplicateur ou le quotient. (CRDP Clermont-Ferrand 1981, p. 7)

Par exemple examinons la division 25 ÷ 6. L'opérateur effectue, à l'aide de la Pascaline, les soustractions successives en inscrivant successivement sur la rondelle le nombre (n) de soustractions effectuées : 25 – 6 = 19 (1) ; 19 – 6 = 13 (2) ; 13 – 6 = 7 (3) ; 7 – 6 = 1 (4). Il s'arrête quand la soustraction n'est plus possible. Il peut lire les deux entiers 1 et 4, reste et quotient de la division euclidienne. L'opérateur utilise directement le résultat d'une soustraction, encore visible sur les lucarnes, pour anticiper la possibilité d'exécuter la soustraction suivante. Seul le nombre 6 est à entrer à chaque fois.

En guise de conclusion sur la Pascaline
- Son fonctionnement repose sur les algorithmes décimaux des opérations sur les entiers naturels, connaissances stables et fondement des calculs numériques de l'époque. Ces algorithmes sont mathématiquement aménagés afin de pouvoir mécaniser les calculs.
- Le reporteur permet à cette machine de réaliser automatiquement l'addition à retenue.
- Cette machine possède des mémoires mécaniques : les *lucarnes* pour enregistrer le premier nombre de l'opération, puis pour afficher les résultats intermédiaires et/ou le résultat final de l'opération ; les sautoirs matérialisant les retenues de l'addition des chiffres ;
- Par contre, il reste à la charge de l'opérateur un certain nombre de tâches de calcul et de mémorisation, qui sont autant d'occasions d'erreurs humaines : calcul mental des compléments à 9 d'un nombre, mémorisation de ces compléments pour les entrer dans la machine, entrer le nombre d'opérations successives dans le cas de la division ou de la multiplication.
Mais la machine de Pascal reste essentiellement une additionneuse.

I.1.2. La machine de Leibniz et la mécanisation de la multiplication

Leibniz veut mécaniser les quatre opérations arithmétiques pour se libérer du calcul, qualifié par lui-même, de travail d'esclave.

> il est indigne d'homme remarquable de perdre des heures à un travail d'esclave, le calcul, qui pourrait fort bien être confié à n'importe qui, avec l'aide de machine. (Leibniz cité par Ligonnière 1987, p. 41)

Ayant découvert la Pascaline lors de son séjour à Paris (1672-1676), il pointe quatre problèmes à surmonter afin de mécaniser la multiplication dans lesquels le problème de la mémoire est central comme le décrit Ligonnière :

> 1- Pouvoir « enregistrer » un nombre avant de le traiter et conserver ce nombre initial, après son traitement.

> 2- Pouvoir commander, par la rotation d'une seule manivelle, des additions successives du nombre enregistré.
> 3- Réaliser la mobilité du chariot additionneur, vers la gauche (multiplication) ou vers la droite (division) en liant ces déplacements à la valeur des divers ordres : unités, dizaines, centaines, milliers, etc.
> 4- Différencier les deux sens de rotation de la manivelle, de façon à effectuer soit des additions-mutiplications, soit des soustractions-divisions à partir d'un seul et même organe inscripteur. (Ligonnière 1987, p. 41)

Leibniz imagine un organe essentiel pour résoudre ces problèmes, le tambour à dents inégales combiné à l'utilisation d'un chariot mobile.

> Suivant la position qui lui était assignée par « l'inscripteur », ce tambour engageait 0, 1, 2…9 de ses cannelures dans le rouage opérateur qui exécutait les calculs. Grâce à cette disposition originale, il était possible de « poser » un nombre, puis de le multiplier par des rotations répétées de la manivelle principale de la machine de Leibniz. (Ligonnière 1987, p. 43)

Il doit cependant patienter 21 ans avant de voir sa machine construite car la fabrication ce tambour à dents inégales exige une précision mécanique très rigoureuse. Les parties essentielles de la machine sont l'inscripteur à 8 chiffres, les lucarnes des résultats, le moniteur de rotation, la manivelle à deux sens, le chariot mobile. Nous expliquons le fonctionnement de cette machine au travers d'un exemple, la multiplication 25×43.
Les chiffres 5, 2, sont entrés respectivement sur l'inscripteur des unités et des dizaines. Les lucarnes n'affichent encore que des zéros. À l'aide d'un style, l'opérateur fait pivoter le moniteur de rotation jusqu'à la position 3 (9 positions de 1 à 9 possibles) : il tourne alors la manivelle jusqu'à son blocage à la fin de la 3^{e} rotation (addition réitérée 25 + 25 + 25). Le résultat 75 s'affiche sur la lucarne de résultats. Le chariot mobile est déplacé d'une position sur la gauche, celle des dizaines. On fait pivoter de nouveau le moniteur de rotation jusqu'à la position 4. L'opérateur tourne la manivelle jusqu'à son blocage à la fin de la 4^{e} rotation (addition réitérée 25 + 25 + 25 + 25 en position des dizaines). Le nouveau résultat 1075 s'affiche sur les lucarnes. Le chariot est déplacé pas à pas par l'opérateur, autant de pas que le multiplicateur possède de chiffres.
La propriété mathématique sur laquelle se fonde la mécanisation de la multiplication est la suivante :

Pour tout nombre naturel $\overline{a_n a_{n-1} ... a_1 a_0}$ on a :

$\overline{a_n a_{n-1} ... a_1 a_0} = a_n \times 10^n + a_{n-1} \times 10^{n-1} + \ldots + a_1 \times 10^1 + a_0 \times 10^0$

D'où, pour tout nombre naturel A et $\overline{a_n a_{n-1} ... a_1 a_0}$ on a :

$$A \times \overline{a_n a_{n-1} ... a_1 a_0} = A \times a_0 + A \times \overline{a_1 0} + ... + A \times \overline{a_{n-1} 0...0} + A \times \overline{a_n 0...00}$$

Leibniz apporte, avec la réalisation de cette machine, des améliorations à la conception des calculateurs mécaniques :

> Inscripteur, viseur de pose, entraîneur, chariot mobile, tambour à des dents inégales, tous ces éléments seront repris dans la plupart des machines suivantes. L'Allemand a été le véritable précurseur du calcul mécanique de bureau. (Ligonnière 1987, p. 43)

Il diminue le nombre de tâches de l'opérateur en matérialisant un nouveau type de mémoire le moniteur de rotation qui stocke un nombre entre 1 et 9 pour la multiplication (ou la division). Les mémoires « lucarnes » et « inscripteur » sont séparées.

Le problème de la mécanisation des quatre opérations est résolu en 1821 par Thomas de Colmar, qui conçoit l'Arithmomètre en filiation avec la Pascaline et la machine de Leibniz. Comme dans la machine de Leibniz, les dispositifs qui servent à inscrire les deux nombres pour l'opération et à afficher le résultat sont dissociés.

I.2. La machine à différences finies et le problème de tabulation d'une fonction numérique

Au début du XIX° siècle, les machines arithmétiques sont utilisées dans les calculs, spécialement dans la fabrication des tables numériques qui outillent les calculs de beaucoup de domaines. Par exemple en 1791, Prony directeur de Cadastre de France doit :

> dresser de nouvelles tables logarithmiques et trigonométriques, nécessitées par la création du système métrique et la division centésimale du quart de cercle. (Ligonnière 1987, p.71)

De nouvelles méthodes mathématiques liées à l'évolution des mathématiques émergent et vont faire évoluer la conception des machines arithmétiques. La méthode des différences finies, utilisant les dérivées successives d'une fonction polynomiale, va permettre d'organiser le calcul des valeurs de toute fonction polynomiales et par approximation celles d'une classe plus large de fonctions.

> La méthode des différences finies peut être appliquée à toutes les fonctions polynomiales […]. Les polynômes sont utilisés dans de nombreuses relations, en physique et en mécanique, et ils permettent de calculer par approximation la valeur d'autres fonctions comme les logarithmes ou les fonctions trigonométriques. (Swade 1993, p. 82)

Nous analyserons de façon détaillée la méthode des différences finies dans le chapitre C1. Nous contentons ici de montrer l'exemple de la tabulation de la fonction « $8x^3 + 6x^2 - 3x - 1$ ».

Nombres x	Images de fonction	Différences premières	Différences deuxièmes	Différences troisième
1	10	71	108	48
2	81	179	156	
3	260	335	204	*48*
4	595	539	*252*	
5	1134	*791*		
6	*1925*			

Dans cet exemple, après avoir calculé les 5 premières valeurs et la 1ère différence finie, on peut constituer la table entière presque mécaniquement en exécutant des additions. (cf. Ligonière 1987)

Babbage, mathématicien anglais, veut mécaniser les tâches répétitives de réalisation de ces tables afin de diminuer le risque d'erreurs dues au calcul humain et au report humain des résultats.

> Babbage espérait que les calculateurs mécaniques seraient parfaitement fiables. Non seulement ses machines effectueraient des calculs rigoureusement exacts, mais elles éviteraient aussi les erreurs de transcription et d'impression des tables mathématiques, en imprimant directement leurs résultats sur des bandes de papier ou sur des plaques métalliques d'impression ; automatiser le calcul et l'impression éliminerait la principale source d'erreurs : intervention humaine. (Swade 1993, p 78)

Cette machine a des capacités étonnantes, puisqu'elle :

> […] peut calculer des valeurs de fonctions polynomiales de degré 7 pour des nombres comportant jusqu'à 31 chiffres (Swade 1993, p. 82)

Cependant, Durand-Richard précise que :

> Babbage affirme, à tort que cette machine aux différences peut effectuer toute espèce de calcul. Si c'est là sans doute une assertion trop optimiste, la machine est néanmoins capable d'obtenir toute fonction récursive primitive[1], dont on peut établir la calculabilité[2] par machine, telle qu'elle est aujourd'hui définie. (Durand-Richard 2003)

[1] Une fonction récursive primitive peut être définie, à partir d'une des fonctions (constantes, successeur ou projection), au moyen de deux procédés standard de construction : la composition et le schéma de récurrence ou de récursion primitive. (cf. http://www.liafa.jussieu.fr/~carton/Enseignement/Complexite)

[2] On dit qu'une fonction est calculable si on peut la calculer (c'est-à-dire si on peut l'évaluer pour toute valeur de son domaine de définition) en utilisant une machine de Turing *ad hoc*. (cf. Hopcroft 1986)

La seule innovation par rapport aux machines précédentes du projet de machine à différence[1] est d'imaginer l'impression automatique des résultats[2]. Le dispositif de sortie qu'est l'imprimante soulage le travail de l'homme dans le report de résultat. Ce dispositif est, en quelque sorte, l'embryon de dispositif de sortie dans la machine ordinateur actuelle

I.3. Conclusion

La figure 3 ci-après schématise l'architecture commune des machines arithmétiques.

L'opérateur humain doit entrer les nombres dans la mémoire - inscripteur, *mémoriser les résultats intermédiaires* à l'aide de rondelles ou d'un moniteur de rotation, faire tourner chaque roue et enfin lire les résultats le résultat à travers les chiffres apparaissant dans les lucarnes pour les copier. L'intervention humaine est donc nécessaire tout au long du processus de calcul. Babbage améliore l'architecture des machines arithmétiques par l'invention de l'imprimante qui supprime l'action de report du résultat.

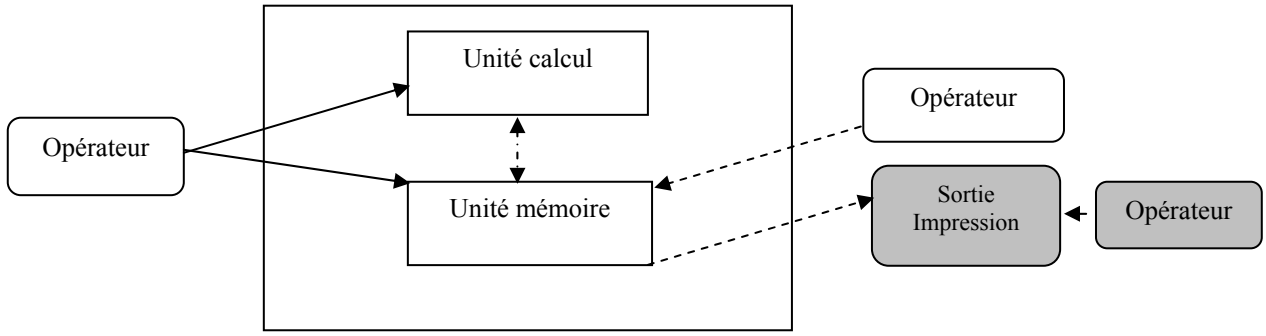


Figure 3. Architecture des machines arithmétiques

Légende : ⟶ : instruction à la machine ; - - ▸ : Transfert d'information
En grisé : nouveauté de la machine à différence finie

Chaque machine arithmétique est en quelque sorte un programme mécanisé : on ne peut changer un programme qu'en changeant de machine. Moreau (1987) qualifie ces machines de *machines à programme figé*.
Si la machine à différences finies aide l'homme à ne plus intervenir durant un calcul, elle ne peut exécuter qu'un seul type de calcul à chaque fois. Ceci est du notamment au principal défaut architectural comme le souligne Ifrah :

> Comme les jacquemarts, les horloges astronomiques et les premiers carillons automatiques, leur organe de commande et leur mécanique d'exécution étaient confondus. Autrement dit, leur dispositif de guidage des opérations, intégré alors aux mécaniques internes, n'était pas indépendant de la structure matérielle de la machine. (Ifrah 1994, p. 535)

C'est en voulant que la machine exécute automatiquement non seulement une tâche de calcul mais *une chaîne de calculs sans intervention de l'opérateur* que Babbage pense à un nouveau projet plus ambitieux, celui de la machine Analytique.

[1] Le mot anglais « difference engine » devrait être traduit en « machine à calculer aux différences finies ». Pour la commodité, nous utilisons, malgré l'inexactitude de la grammaire et de logique, le mot « machine à différence ».
[2] En 1991, lors du bicentenaire de la naissance de Babbage, la machine à différence N°2 est exposée au musée des Sciences à Londres : elle « constitue la pièce maîtresse de l'exposition *Charles Babbage et la naissance de l'ordinateur* » (Swade 1993, p. 84).

II. Une première rupture : la machine Analytique de Babbage

L'invention des métiers à tisser par Jacquard au XVIIe siècle marque un pas important dans la communication entre l'homme et la machine par l'introduction des programmes extérieurs à l'aide de cartons perforés « préfigurant ainsi la notion moderne de *programme* » (Ifrah 1994, p. 536).
Le champ des algorithmes de calculs s'élargit, résultant des avancées en analyse et en algèbre comme par exemple, l'invention du calcul différentiel et intégral (Leibniz en 1672), la solution d'un système d'équation (Cramer en 1750), la théorie des fonctions analytiques (Lagrange en 1797).
Ce nouveau contexte technologique et mathématique va offrir à Babbage les moyens de concevoir une machine capable de traiter mécaniquement une chaîne de tâches de calcul.

II.1. La machine de Babbage, une machine à programme externe

L'invention de Jacquard rend désormais possible d'envisager des *programmes extérieurs* à l'aide de cartons perforés pour commander une chaîne d'actions à une machine. Ce sont des *machines à programme externe* (Moreau 1987).
Cette invention ainsi que les réflexions sur les machines à différence permettent à Babbage de concevoir le projet de la machine Analytique, machine capable d'exécuter plusieurs opérations successives à l'aide d'un programme qui est une séquence d'instructions inscrites sur des cartes perforées.
La longue citation suivante présente d'abord les composants hérités des précédentes machines - report de retenues, parties arithmétiques - , mais aussi les limites des machines précédentes et les solutions recherchées qui contiennent en germe les trois unités principales d'un ordinateur moderne : mémoire, unité de commande, unité arithmétique.

> - Pour éviter toute intervention humaine : lorsque la différence constante était sujette à des changements périodiques, on pouvait envisager que *la machine réalise d'elle-même les modifications nécessaires* ;
> - Babbage se rend compte également que, pour que la machine dispose d'un plus grand éventail de possibilités de calcul, il suffirait *qu'elle soit commandée* par un mécanisme entièrement indépendant de la partie additionneuse ;
> - Afin que d'autres opérations que l'addition puisse être effectuées, la multiplication par exemple, et que *l'enchaînement des opérations ne soit pas rigidement commandée par l'architecture de la machine* ;
> - Le problème des retenues, d'abord effectuées l'une après l'autre, impose, pour réduire le temps de calcul, de penser la simultanéité et conduit à poser celui de la mémoire, qui induit la nécessité du *stockage* et du *transport* ; la complexité du mécanisme des retenues ainsi introduit interdit qu'il soit reproduit à tous les étages de la machine, ce qui conduit Babbage à l'idée d'un *mécanique arithmétique unique et central*, accomplissant directement l'une quelconque des quatre opérations, dans un ordre indiqué par un *organe de commande* ;
> - Pour résoudre certaines équations aux différences, telles que $\Delta^7 u_x = u_x$ ou $\Delta^7 u_x = a\Delta u_x$ Babbage est amené à concevoir une organisation circulaire, qui fait l'intervenir le résultat sur la marche ultérieure du calcul de manière rétroactive. (Durand-Richard 2003, souligné par nous)

Babbage conçoit principalement 3 types de cartes permettant de programmer la machine :
- cartes d'opérations arithmétiques et de transfert
- cartes nombres : constantes et variables
- cartes combinatoires.

Grâce à ces cartes, sa machine est capable de choisir les opérations à effectuer, soit à partir d'une règle qui lui a été donnée, soit en fonction du résultat des calculs précédents. Une telle possibilité, connue sous le terme de « branchement » ou de « saut conditionnel » permet à la machine analytique de poursuivre seule sa tâche, après sélection de la décision appropriée.

> L'apparition de cette faculté de décision constitue l'une des caractéristiques principales de la machine Analytique. Elle la distingue catégoriquement de toutes les machines à calculer mécaniques précédentes

comme de bien d'autres qui suivront encore et qui doivent toujours être guidées, pas à pas, par l'utilisateur. (Ligonnère 1987, p. 104)

II.2. Architecture de la machine analytique

En dehors de la faculté de décision, Babbage conçoit pour la première fois la notion de mémoire (« store »). Cette machine possède donc certaines des caractéristiques des ordinateurs modernes :

- une architecture qui annonçait celle de tous les calculateurs universels puisqu'elle comprenait :
1. Une unité arithmétique et logique ; 2. Une mémoire ; 3. Une unité de commande ; 4. Une unité d'entrée ; 5. Une unité de sortie,
- utilisait des cartes perforées comme support de communication, support dont le rôle est encore aujourd'hui des plus importants ;
- était numérique, montrant ainsi la voie aux calculateurs et ordinateurs. (Moreau 1987, p. 21)

En nous appuyant sur les travaux de Moreau 1987 et Durand-Richard 2003, décrivons rapidement chaque partie de la machine (voir aussi une représentation de cette machine : figure 5) :

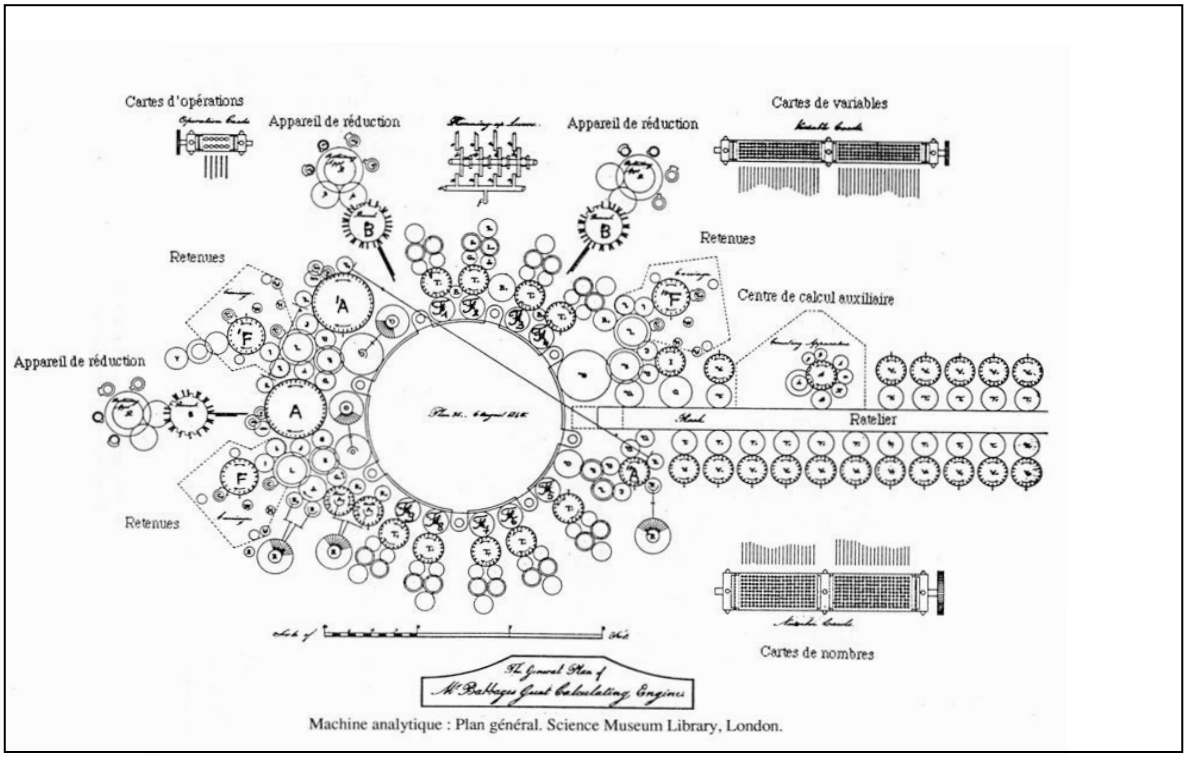


Figure 4. Une représentation de la machine Analytique (d'après le « Science Museum Library » de Londres)

Mémoire (The Store)

Elle est structurée en colonnes composées d'axes verticaux, porteurs de rouages décimaux, qui emmagasinent des nombres pour les calculs à venir. Chaque nombre est transféré d'un axe primaire à un axe secondaire d'où il est prélevé par de longues crémaillères qui l'acheminent jusqu'à l'unité centrale de calcul. Et un transfert analogue a lieu dans l'autre sens après le calcul, sur des crémaillères différentes. Les transferts entre mémoire et unité de calcul sont dirigés par une carte insérée dans un cadre situé sous le rouage le plus bas, et qui désigne la nature de cette variable, ce qui correspond à l'existence d'un programme indiquant soit le nombre à lire, soit l'emplacement où l'inscrire.

L'enregistrement et l'usage d'un nombre dans la mémoire s'effectuent en utilisant une propriété d'effaçabilité qui permet à une même colonne de contenir successivement des nombres différents

> En effet, le fait de prélever un nombre dans la mémoire l'effaçait ; un deuxième jeu, situé dans la même partie supérieure du même axe, constituait donc une copie du nombre enregistré à l'origine. […] Pour utiliser un nombre enregistré dans la mémoire, on le transférait d'abord d'un axe primaire à un axe secondaire d'où il était prélevé par des longues crémaillères qui l'acheminaient jusqu'à l'unité de calcul. Pour mémoriser un nouveau nombre, issu de calculs, on effectuait les mêmes opérations en sens inverse. (Ligonnière 1987, p. 97)

La machine Analytique de Babbage se distingue des machines arithmétiques précédentes par cette caractéristique fondamentale de la mémoire qui participe, avec la nouvelle capacité de décision, à l'élargissement des capacités de calcul.
Dans la mémoire se trouve également un dispositif de calcul auxiliaire, qui permet de réaliser certains calculs intermédiaires simples, par la méthode des différences, sans avoir à envoyer les nombres dans l'unité de calcul (The Mill). Il s'articule sur un des axes à chiffres du ratelier, et sur l'axe de report de l'unité de calcul.

Unité de calcul (The Mill)

C'est l'unité arithmétique où sont effectuées les opérations. Elle est organisée circulairement.
Chaque calcul est codé par une suite d'instructions, donnée sur une suite de cartes perforées.
Elle est essentiellement constituée de :
- 3 axes de chiffres assurant les multiplications et divisions par 10 ;
- 3 axes de report des dizaines pour les additions, dont l'un étant connecté à la fois avec le moulin et le magasin ;
- 9 axes constituant une table de multiplication ;
- et un appareillage de sélection de ces multiplications situé sous ces axes de tables.

Cette unité contient par ailleurs un levier « run-up » qui matérialise la comparaison avec zéro :

> There are certain functions which necessarily change in nature when they pass through zero or infinity, or whose values cannot be admitted when they pass these limits. When such cases present themselves, the machine is able, by means of a bell, to give notice that the passage through zero or infinity is taking place, and it then stop until the attendant has again set it in action for whatever process it may next be desired that shall perform. (Menabrea 1842)

Unité de commande

Ce dispositif gère des opérations à effectuer dans l'unité de calcul. Il est constitué de 3 cylindres, formés chacun de 50 ou 100 lames métalliques plates, chaque lame pouvant recevoir 4 ergots ou pignons, et qui donnent ainsi 200 à 400 positions pour codifier les instructions de traitement des opérations. La position des ergots sur chaque cylindre, ainsi que la rotation de chaque cylindre sur son axe, et leur mouvement d'avancée ou de recul latéral, permettent de dicter la nature des opérations à effectuer, et le moment où le faire.
Le mouvement d'un cylindre peut ainsi commander :
- l'entrée ou la sortie d'un nombre de l'axe d'entrée dans le moulin, nombre qui alors peut être effacé ou retenu dans le moulin,
- l'avancée d'une carte de variable, ou d'opération,
- la rotation d'un cylindre, ou d'un autre, sur son axe.

Registre compteur (counting apparatus)

La machine contient un *registre compteur* pour enregistrer le nombre d'opérations élémentaires effectuées par la machine. Ce dispositif est nécessaire dans les séquences répétitives de calcul.

Lecteur des cartes perforées

Ce lecteur de carte, dispositif emprunté au métier à tisser de Jacquard, est matérialisé par un système qui se compose d'un prisme porteur de carte, de biellettes, et de tiges.

Nous schématisons l'architecture de la machine Analytique de Babbage dans la figure 4 ci-après.

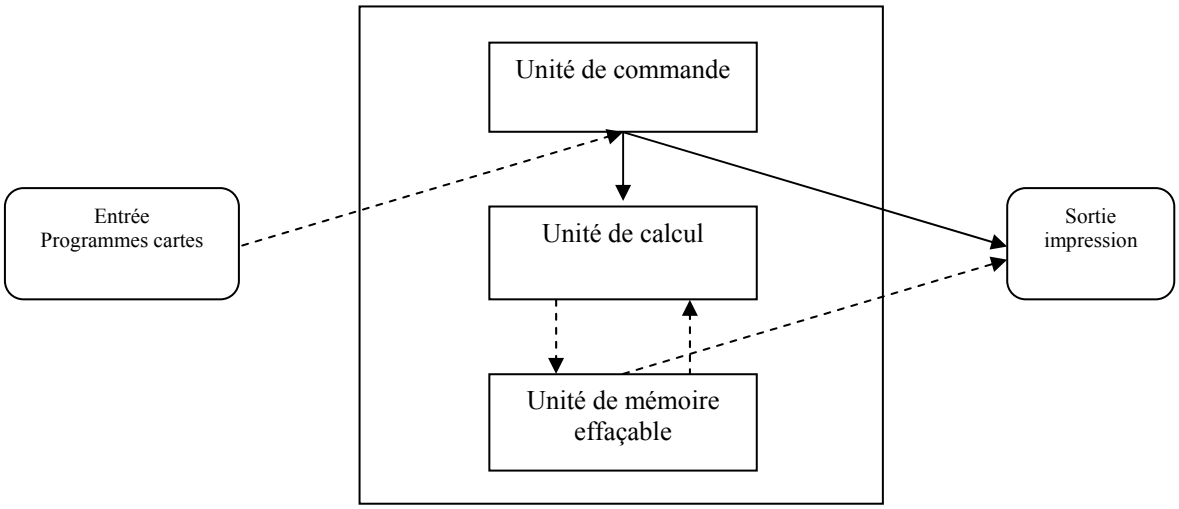


Figure 5. Architecture de la machine Analytique

Légende : ⟶ : Instructions ; - - ▸ : Transfert d'information

II.3. Émergence des notions de boucles et de variables

Un programme pour cette machine est donc composé d'une chaîne des cartes de contenus différents et variés (cf. Walker 2003). Plusieurs programmes ont été écrits pour cette machine par la mathématicienne Ada (1840), dans lesquels on peut attester de la *première apparition des notions de cycle (boucles), de « subroutines » (sous-programmes) et de variables comme objets non mathématiques.*

Pour appuyer notre affirmation, examinons le processus d'écriture du programme pour le calcul des nombres Bernoulli :

Partant de la formule connue où B_i désigne les nombres Bernoulli à calculer :

$$\frac{x}{e^x - 1} = 1 - \frac{x}{2} + B_1 \frac{x^2}{1 \times 2} + B_3 \frac{x^4}{2 \times 3 \times 4} + B_5 \frac{x^6}{2 \times 3 \times 4 \times 5 \times 6} + \ldots$$

Ada déduit une formule de récurrence :

$$-\frac{2n-1}{2(2n+1)} + B_1 \frac{2n}{2} + B_3 \frac{2n \times (2n-1) \times (2n-2)}{2 \times 3 \times 4} + \ldots + B_{2n-1} = 0$$

Soit

$A_0 + A_1B_1 + A_3B_3 + \ldots B_{2n-1} = 0$.

Cette formule lui permet de :

- transformer une formule contenant *une infinité dénombrable de valeurs en un processus de calcul n-fini*.

Ce processus est basé sur la formule de récurrence double : A_{2i+1}en fonction de A_{2i-1} ($i \geq 1$) et B_{2i+1} en fonction de B_{2j+1} et de A_{2j+1} ($j = 1 \ldots i-1$)

- attacher la condition d'arrêt du processus de calcul à la valeur finie n.

Ada écrit alors un programme pour n = 4 et l'exécute "à la main".

Nous voyons donc que *l'écriture du programme s'accompagne d'un retour réflexif sur les objets mathématiques de la formule afin de dégager la condition d'arrêt et l'invariant d'une boucle*.
Dans ce programme destiné à la machine analytique de Babbage (non construite, rappelons-le), seule la répétition n fois d'une série d'instructions (la boucle For) est possible, la condition étant de la forme « si … faire » (si X = 0, faire A sinon faire B). (Walker 2003, Ligonnière 1987).
Nous examinerons de façon plus détaillée le programme d'Ada dans le chapitre B3 de cette partie

II.4. Conclusion

L'existence d'un programme externe séparé de la machine va permettre potentiellement à la machine de Babbage d'accroître les possibilités mécaniques de calcul.

> [Elle peut] calculer le développement de n'importe quelle fonction, donc n'importe laquelle des fonctions alors connues Elle constitue donc une sorte de matérialisation, de traduction de l'analyse. (Durand-Richard 2003).

D'où le nom de machine Analytique.

> Babbage l'a clairement senti lorsqu'il écrivait que sa machine, elle, pourrait permettre de résoudre n'importe quelle équation et exécuter les opérations les plus compliquées de l'analyse. Son intuition recevra un fondement théorique en 1936, lorsque Turing montra que toute fonction calculable pouvait l'être à l'aide d'une unique machine numérique, abstraite certes, mais dont la machine de Babbage était une préfiguration. (Moreau 1987, p. 20)

De plus, la mémoire de cette machine a pour propriété fondamentale d'être effaçable contrairement à la mémoire des machines arithmétiques.
Dans le chapitre suivant (B3), nous montrerons comment Ada dans le premier programme à cette machine utilise ce caractère d'effaçabilité de la mémoire pour définir la notion de variable.
Mais cette machine à programmes externes a des limites.
Les instructions, donc les programmes, sont contenues dans un dispositif externe (cartes perforées) que l'unité de commande vient lire pas à pas pour exécuter les instructions successives qui y figurent. Si un programme doit être exécuté de nouveau, il faut qu'un opérateur humain l'introduise de nouveau dans la machine, ce qui allonge le temps du traitement.
De plus, il y a une séparation totale entre l'organe de commande (programmeur à cartes contenant les ordres de commande) et les autres organes et informations, en particulier les données et résultats du calcul. *Absentes des mémoires, les instructions une fois exécutées disparaissent pour la machine*.

> Cette idée que le résultat d'une action puisse réagir sur une commande a émergé lentement au cours du XIXème siècle. Par contre il faudra attendre le XXe siècle pour qu'on se permette de traiter automatiquement les ordres de commande comme de vulgaires données. (Verroust 2004)

Nous allons voir brièvement comment Von Neumann a dépassé cette limitation fondamentale.

III. Une deuxième rupture : la machine de Von Neumann

La contribution fondamentale de Von Neumann à la conception de la machine ordinateur est nommée couramment « Principe de Von Neumann ». Elle est une réponse aux limitations de la machine analytique : selon ce principe, les instructions doivent être contenues dans la mémoire et la machine doit fonctionner sur programme enregistré. *Les instructions peuvent alors être modifiées par la machine durant l'exécution même d'un programme*.

L'idée de programme enregistré présente deux avantages principaux, celui de l'accélération des calculs et celui de la modification des instructions par la machine elle-même :

> Un premier [avantage] résultait de l'accélération des calculs. […] Un autre avantage, essentiel celui-là, résultait de ce que les instructions pouvaient désormais être traitées comme des données. (Moreau 1987, p. 41)

Et en ce sens cette machine se différencie fondamentalement des machines calculateurs et prend le nom d'ordinateur :

> On appelle **ordinateur** un calculateur universel à programme enregistré. (Moreau 1987, p. 41)

III.1. Architecture de la machine de Von Neumann

Pour expliquer la conception de la machine EDVAC, Von Neumann décrit comme suit l'architecture de sa machine, prototype de l'ordinateur moderne :

> Ce rapport organise le système d'un ordinateur en quatre principales parties : l'unité arithmétique centrale (AC), l'unité centrale de commande (CC), la mémoire (M) et l'entrée et sortie composant (IO). L'AC a pour but d faire des opérations arithmétiques de base, peut-être d'un niveau plus élevé pour des fonctions arithmétiques comme radical, logarithme, fonctions trigonométriques, et leur fonction inverse. Le CC est de commander des propres séquences d'opérations et supervise chaque unité à exécuter ensemble pour une tâche programmée du système. La mémoire est pour stocker en même temps des données numériques (valeurs initiales, valeurs constantes, tables ou des fonctions fixées) et des instructions numériquement codées. Et l'unité IO sert de l'interface de l'ordinateur. (Barney et Cabrera 2003)

En suivant un ordre de présentation similaire à celui de la machine analytique, nous décrivons maintenant les parties essentielles d'un ordinateur (référence Discala 2004, pp. 41-47):

Unité de mémoire

Les données et les résultats intermédiaires ou finaux, les programmes sous forme d'instructions, sont contenus dans une zone de la mémoire. Les instructions peuvent elles-mêmes faire l'objet de traitements comme les données. La mémoire effaçable, depuis la machine Analytique de Babbage, occupe le premier rôle par rapport aux possibilités de calcul, et donc dans l'architecture de l'ordinateur :

> L'ordinateur était défini comme un calculateur universel (general purpose computer) composé de plusieurs parties, chacune spécialisée dans l'accomplissement de tâches précises : les opérations arithmétiques et logiques, la mémoire, le contrôle et l'interaction avec un opérateur humain. L'automatisme consiste en ce qu'un calcul, une fois commencé, doit aller jusqu'à sa conclusion sans intervention humaine. Les instructions doivent être contenues dans la mémoire mais elles ne doivent pas faire partie de la structure de la mémoire ; la machine doit donc fonctionner sur programme enregistré. De ce fait, la mémoire assumait le rôle de premier plan. (Ramunni 1989, p. 65)

Unité de calcul (unité de traitement)

Elle effectue les opérations élémentaires de types arithmétiques ou booléennes nécessaires à l'exécution d'un programme. Elle contient un registre de données (RD), un accumulateur (ACC) et une unité arithmétique et logique (UAL).

Unité de commande (unité de contrôle)

Elle est chargée de commander et de gérer les différents constituants de l'ordinateur :

+) Un compteur ordinal ou compteur-instruction (CO) qui contient l'adresse dans la mémoire centrale de l'instruction à exécuter,

+) Un registre-instruction (RI) qui reçoit une instruction élémentaire de programme à exécuter ;

+) Un registre adresse RA qui contient l'adresse de la prochaine instruction à exécuter ;

+) Un décodeur de fonction, associé au RI qui analyse l'instruction à exécuter et entreprend les actions appropriées dans l'UAL ou dans la mémoire ;
+) Une horloge.

L'unité de commande vient chercher les instructions élémentaires, les décode et en commande l'exécution dans l'ordre prescrit assurant le déroulement séquentiel des opérations. Les instructions du programme contenues dans la mémoire sont lues par cette unité. L'unité de commande et l'unité de calcul constituent l'unité centrale ou le processeur central.
Nous schématisons dans la figure 5 ci-après l'architecture de la machine ordinateur de Von Neumann :

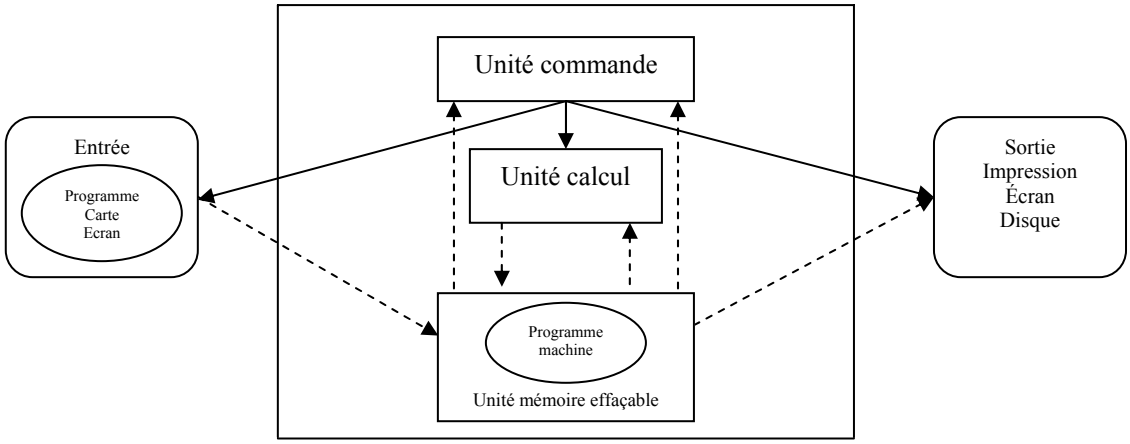


Figure 6. L'architecture de la machine ordinateur de Von Neumann

Légende : ⟶ : Instructions ; ⇢ : Transfert d'information

Les avancées théoriques en mathématiques et en physique (travaux de Turing, Von Neumann etc.), et les avancées technologiques (registres, systèmes de communication etc.), permettent en 1948 de fabriquer aux États-Unis « la machine de Manchester », première machine ordinateur entièrement électronique et construite conformément au plan de Von Neumann .

III.2. La puissance de calcul de la machine de Von Neumann

En travaillant sur le 10^{e} problème de Hilbert, Turing a essayé de formuler la réponse à la question de l'existence d'un algorithme universel pour une classe de problèmes et de trouver un modèle théorique de la machine ordinateur.
Un problème est réputé décidable lorsqu'il existe un algorithme dont l'entrée est une instance du problème et dont la sortie est une réponse au problème. Les années 30 ont consacré l'existence de problèmes indécidables ce qui a nécessité de définir la notion d'algorithme.

> Pour montrer qu'il n'existe pas d'algorithme, ou même l'énoncer rigoureusement, nous avons besoin d'une définition de ce qu'est un algorithme. (Matiiassévitch 2000)

Des logiciens et des mathématiciens comme Gödel, Church, Kleene, Post et Turing ont cherché à formaliser la notion d'algorithme. Nous retiendrons la définition attachée à la machine de Turing :

> Turing utilisa une machine hypothétique que nous appelons « machine de Turing ». Turing définissait l'algorithme comme un ensemble d'instructions pour sa machine simple. (Goldschlager et *al.*, 1986)

En effet, les différentes définitions formelles de la notion d'algorithme sont équivalentes :

> Tout ce qui peut être calculé peut l'être par une machine de Turing. (Goldschlager et *al.*, 1986)

La thèse de Church-Turing postule donc que tout problème de calcul basé sur une procédure algorithmique peut être résolu par une machine de Turing. Tout programme d'ordinateur peut donc être traduit en une machine de Turing.
Si on se restreint au problème de la tabulation d'une fonction, quelles sont les fonctions mathématiques calculables par la machine de Von Neumann ? La réponse donnée est que toute fonction pour laquelle il existe un ensemble d'instructions pour la machine de Turing ou encore pour laquelle existe un algorithme est calculable.

IV. Conclusion

Nous venons de survoler le processus qui a conduit à la naissance de l'ordinateur, depuis la machine arithmétique de Pascal, en passant par la machine Analytique de Babbage, jusqu'à la machine ordinateur de Von Neumann. Ces machines sont nées dans le but de résoudre des problèmes mathématiques de calcul.

L'accroissement du champ des problèmes calculables

Un problème calculable par une machine, quelle qu'elle soit, est un problème pour lequel existe une solution liée à un algorithme.
Les machines arithmétiques permettent de mécaniser les opérations arithmétiques en s'appuyant sur l'existence d'algorithmes décimaux.
La machine analytique de Babbage peut calculer (potentiellement) une classe très large de fonctions, celle des fonctions récursives primitives, en s'appuyant sur l'algorithme des différences finies. Ces machines bénéficient à la fois des progrès technologiques (métiers à tisser de Jacquard) et des avancées mathématiques de l'époque en particulier dans le champ de l'analyse.
La machine de Von Neumann peut, quant à elle, calculer des fonctions calculables telles que définies plus haut et réaliser une extrême diversité de tâches.

Le rôle central de l'évolution de la mémoire : effaçabilité

Nous avons vu que la nécessité de réduire au maximum l'intervention de l'homme dans la machine au cours de l'exécution de calcul a fait évoluer la structure technologique de la machine, notamment la mémoire, dans sa genèse comme le montre la figure 7.

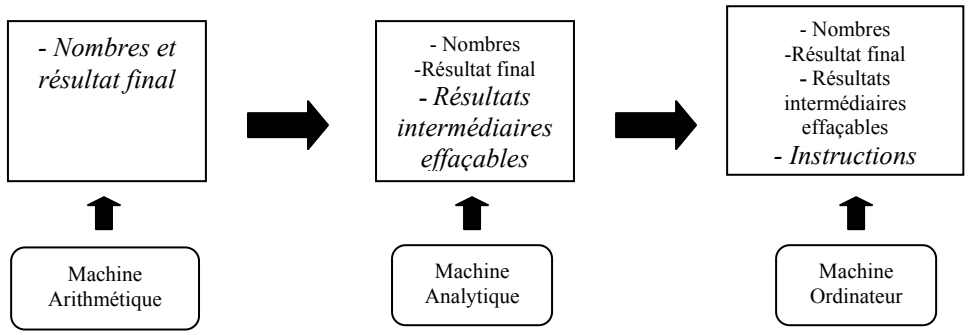


Figure 7. Évolution du stockage de la mémoire selon le type de machine

Le schéma montre clairement que les trois étapes « Machine Arithmétique », « Machine Analytique » et « Machine Ordinateur » sont marquées par l'accroissement des possibilités de stockage qui elles-mêmes accroissent les possibilités de calcul de la machine.
Ces étapes sont caractérisées d'autre part par la propriété fondamentale d'effaçabilité de la mémoire.

L'évolution de la mémoire a permis de transformer les relations entre l'homme et la machine. Avec la possibilité mécanique du stockage en mémoire de résultats intermédiaires liée elle-

même à la propriété d'effaçabilité de ses résultats, l'exécution mécanique de calculs répétitifs est rendue possible (machine analytique de Babbage). Cela a pour conséquence fondamentale l'écriture d'un programme à une machine, la notion d'effaçabilité étant intrinsèquement liée à la notion de variable et de boucle comme nous le montrerons dans le chapitre suivant (chapitre B3). Rappelons que Kuntzmann en avait fait un objet d'enseignement fondamental dans son traité de 1958 (chapitre B1).

Nous donnons une définition de la notion de mémoire effaçable qui servira de référence pour la suite de notre travail :

> Une mémoire effaçable est un emplacement dans l'unité de mémoire où l'on peut intentionnellement stocker une valeur. Ce stockage efface l'ancienne valeur contenue dans cette mémoire.
> L'unité de mémoire d'une machine qui possède un certain nombre de mémoires effaçables est dite effaçable.

Le concept de programme enregistré

Le concept de programme enregistré dans une mémoire commune avec les données (machine de Von Neumann) permet de traiter le programme lui-même comme des données de calculs. Un programme, une fois exécuté, peut devenir une instruction pour un autre programme. Nous schématisons cette évolution dans la figure 8.

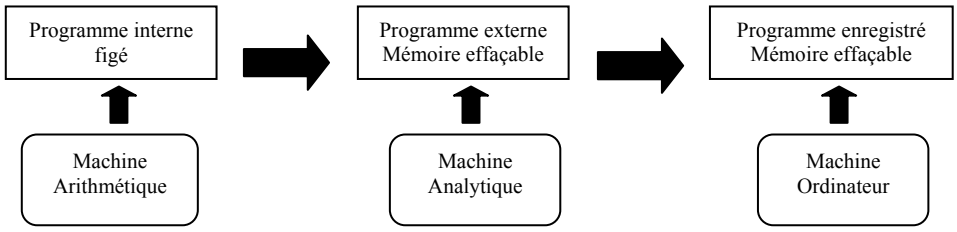


Figure 8. Évolution de la place du programme selon le type de machine

Depuis la naissance du premier ordinateur fondé sur le principe de Von Neumann l'EDVAC en 1946 aux États-Unis, la technologie des composants matériels a connu une évolution fulgurante avec l'invention des transistors, des mémoires à tore de ferrite, puis des circuits intégrés. Cependant le fonctionnement d'un ordinateur reste attaché à un paradigme inchangé (pour reprendre des termes de Baron 1987) qui est celui de l'architecture de Von Neumann : autrement dit l'architecture de tous les ordinateurs existants est celle de Von Neumann.
Les efforts pour faciliter la communication entre l'homme et la machine se reportent alors sur l'amélioration des langages de programmation qu'autorise le programme enregistré.

Comment et pourquoi ces langages ont-ils évolué ?
Quelle la genèse des notions de variable et de boucle dans cette évolution et les raisons de cette genèse ?
Quels liens entretiennent ces langages avec la machine ?

Dans le chapitre suivant, nous allons tenter de donner des éléments de réponses à ces questions.

Chapitre B3
Du programme d'Ada au langage évolué

Introduction

Ce chapitre se compose de deux parties.
Dans la première partie, nous nous intéressons à l'écriture des premiers programmes « informatiques » destinés aux premières machines à mémoire effaçable : machine Analytique de Babbage et machine ordinateur de Von Neumann. Nous examinons tout particulièrement l'émergence des notions de variable et de boucle associées à l'écriture des programmes de calcul répétitif.
Dans la deuxième partie, nous décrivons l'évolution des langages de programmation depuis l'apparition des premières machines ordinateurs en essayant de dégager des règles d'écriture communes dans la formulation des boucles dans des langages évolués.

I. Écriture de programme à la machine Analytique de Babbage

I.1. Notion de boucle comme «cycle d'opérations »

Comme nous l'avons déjà dit (chap. 2, partie B), Ada, mathématicienne anglaise et collaboratrice de Babbage écrit le schéma du premier programme adressé à la machine Analytique de Babbage (non construite). En 1842, parmi les notes qu'elle ajoute à la traduction en anglais d'un article de Menabrea (1842)[1], Ada définit le principe d'itérations successives lorsqu'elle esquisse un programme de calcul des nombres de Bernoulli et introduit alors le concept de *cycle d'opérations*, ancêtre de la notion de boucle :

> Wherever a *general* term exists, there will be a *recurring group* of operation, as in the above example. Both for brevity and for distinctness, a *recurring group* is called a *cycle*. A *cycle* of operations, then, must be understood to signify any *set of operations* which is repeated *more than once*. It is equally a *cycle*, whether it be repeated *twice* only, or an indefinite number of times; for it is the fact of a *repetition occurring at all* that constitutes it such. (Ada 1842, Note E)

Seule la machine Analytique possède des conditions techniques suffisantes (dispositif « run up »[2] et mémoire effaçable) pour permettre ainsi d'envisager la réalisation d'un cycle boucle :

> A method was devised of what was technically designated *backing* the cards in certain groups according to certain laws. […] The object of this extension is to secure the possibility of bringing any particular card or set of cards into use *any number of times successively* in the solution of one problem. (Ada 1842, Note C)

Ada voit aussi dans la notion de cycle un moyen d'économiser des cartes sur lesquelles les programmes sont « écrits » et donc de réduire la longueur du programme à écrire :

> The power of *repeating* the cards, alluded to by M. Menabrea, and more fully explained in note C, reduces to an immense extent the number of cards required. (Ada 1842, Note F)

La seule boucle permise par le dispositif « Run up » est la boucle « Aller à… » avec condition d'arrêt décrémentée.

[1] Cet article résume en français une série de séminaires donnés par Babbage en Italie.
[2] Dispositif de la machine de Babbage qui dès qu'il lit 0 (zéro) déclenche le retour du lecteur de carte à une carte repérée pour répéter un groupe de cartes.

I.2. Notion de variable comme « colonne variable »

Ada définit d'abord les colonnes dans la mémoire comme des *Variables* dans le sens où les valeurs enregistrées sur ces colonnes sont destinées à varier au cours du programme :

> The V's are for the purpose of convenient reference to any column, either in writing or speaking, and are consequently numbered. The reason why the letter V is chosen for the purpose in preference to any other letter, is because these columns are designated (as the reader will find in proceeding with the Memoir) the *Variables*, and sometimes the *Variable columns*, or the *columns of Variables*. The origin of this appellation is, that the values on the columns are destined to change, that is to *vary*, in every conceivable manner. (Ada 1842, Note B)

En distinguant différents types de colonnes suivant la nature des variables, elle fait apparaît l'idée qu'une variable reçoit une valeur.

> In all calculations, the columns of Variables used may be divided into three classes:
> 1st. Those on which the data are inscribed ;
> 2ndly. Those intended to receive the final results ;
> 3rdly. Those intended to receive such intermediate and temporary combinations of the primitive data as are not to be permanently retained, but are merely needed for *working with*, in order to attain the ultimate results. (Ada 1842, Note D)

Ada différencie les « variable columns » selon les 3 étapes chronologiques du processus d'exécution du programme, de l'entrée « variables for data » à la sortie « variables for results ». Seules les variables de la deuxième étape « working variable » participent au processus de calcul et changent de valeurs durant le calcul.

> Combinations of this kind might properly be called *secondary* date. They are in fact so many *successive stages* towards the final result. The columns which receive them are rightly named *Working-Variables*, for their office is in its nature purely *subsidiary* to other purposes. They develop an intermediate and transient class of results, which unite the original data with the final results. (Adad 1842, Note D)

Les colonnes destinées à recevoir les « working variables » sont effaçables, caractéristique importante pour la notion de variable et déjà commentée lors de l'analyse du traité de Kuntzmann (cf. Chapitre 1, partie B).
L'idée que les trois types de variables sont de même nature est présente chez Ada :

> The Result-Variables sometimes partake the nature of Working-Variables. It frequently happens that a Variable destined to receive a final result is the recipient of one or more intermediate values successively, in the course of the processes. Similarly, the Variables for data often become Working-Variables, or Results-Variables, or even both in succession. (Adad 1842, Note D)

Ada introduit une notation des colonnes par un système d'indices pour différencier les différents états d'une variable durant un calcul.

> The *lower* indices are obviously indices of *locality* only, and are wholly independent of the operations performed or of the results obtained, their value continuing unchanged during the performance of calculations. The *upper* indices, however, are of a different nature. Their office is to indicate any *alteration* in the value which a Variable represents; and they are of course liable to changes during the processes of a calculation. Whenever a Variable has only zeros upon it, it is called ^{0}V; the moment a value appears on it (whether that value be placed there arbitrarily, or appears in the natural course of a calculation), it becomes ^{1}V. If this value gives place to another value, the Variable becomes ^{2}V, and so forth. Whenever a *value* again gives place to *zero*, the Variable again becomes ^{0}V, even if it have been ^{n}V the moment before. If a *value* then again be substituted, the Variable becomes ^{n+1}V (as it would have done if it had not passed through the intermediate ^{0}V). Just before any calculation is commenced, and after the data have been given, and everything adjusted and prepared for setting the mechanism in action, the upper indices of the Variables for data are all unity, and those for the Working and Result-variables are all zero. (Ada 1842, Note D)

Une colonne variable est ainsi notée $^{n}V_{m}$ où l'indice m désigne la position de la colonne dans la mémoire, nous le nommons « indice de position » (indice of *locality*). L'indice n désigne

l'état nième de cette colonne variable durant l'exécution du calcul, nous l'appelons indice d'état (*alteration* in the value). Ce dernier indice permet de « suivre » une colonne variable.
Pour se faire comprendre, Ada donne l'exemple du calcul des solutions d'un système linéaire de deux équations à deux inconnues avec l'algorithme de Cramer (figure 1).

$$\begin{cases} mx + ny = d \\ m'x + n'y = d' \end{cases}$$

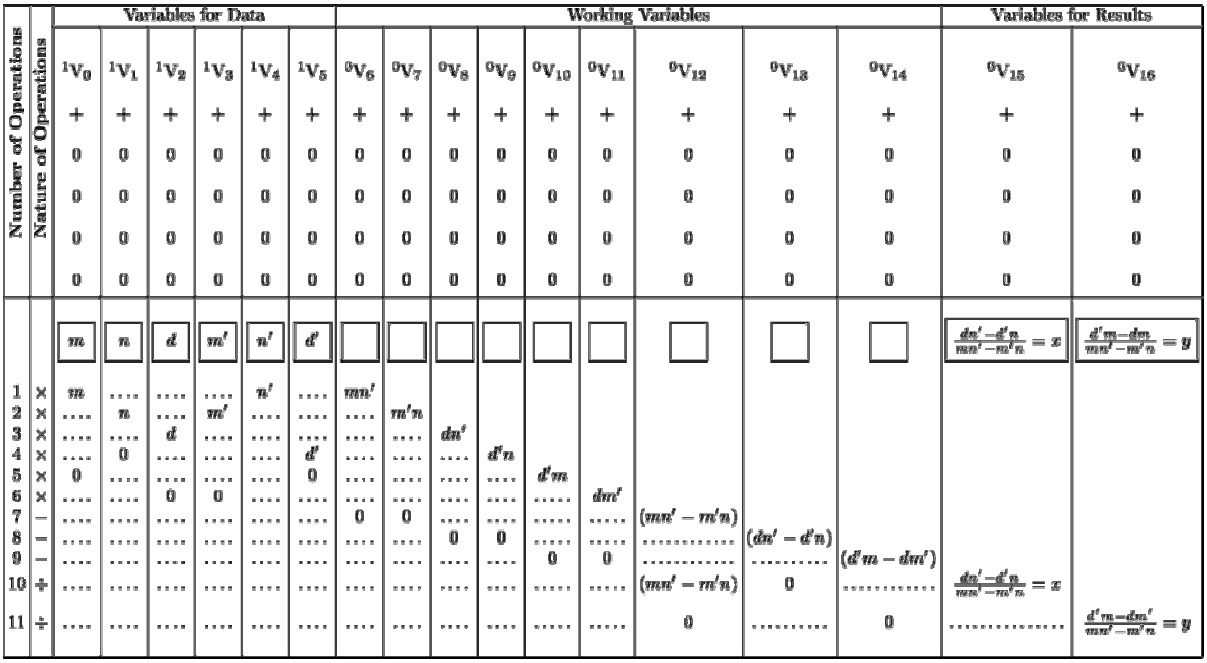


Figure 1. Trois types de variables dans un programme de calcul (reproduction disponible sur http://www.fourmilab.ch/babbage/sketch.htm)

Ce programme appelle quelques commentaires.
Le terme « variable » est utilisé pour des colonnes qui reçoivent des valeurs constantes (m, n, d, etc.) comme on peut le constater pour les colonnes variables de V_0 à V_5. Ces colonnes peuvent, bien entendu, recevoir d'autres valeurs pour d'autres équations, mais elles ne changent pas de valeurs durant le calcul.
Toutes les colonnes sont mises à zéro dès qu'elles ne participent plus au processus de calcul.
L'enregistrement des résultats intermédiaires dans des colonnes variables est actualisé à chaque opération. Le nombre de colonnes des « working variable » et celui des valeurs intermédiaires doivent être égaux.
Les explications, la codification par un système d'indices et la valeur zéro ainsi que la numérotation de chaque étape permettent ainsi le suivi de l'exécution pas à pas du programme-carte par la machine. On a déjà là les germes d'un programme informatique dans ses relations avec une machine

Durant un calcul, les valeurs enregistrées dans une colonne « working variables » changent. Comment Ada note t-elle ce changement d'une même colonne ?

> There are several advantages in having a set of indices of this nature; but these advantages are perhaps hardly of a kind to be immediately perceived, unless by a mind somewhat accustomed to trace the successive steps by means of which the engine accomplishes its purposes. We have only space to mention in a general way, that the whole notation of the tables is made more consistent by these indices, for they are able to mark a *difference* in certain cases, where there would otherwise be an apparent *identity* confusing in its tendency. In such a case as $V_n = V_p + V_n$ there is more clearness and more consistency with the usual laws of algebraical notation, in being able to write ${}^{m+1}V_n = {}^{q}V_p + {}^{m}V_n$. It is also obvious that the indices furnish a powerful means of tracing back the derivation of any result; and of registering

> various circumstances concerning that *series of successive substitutions*, of which every *result* is in fact merely the final consequence; circumstances that may in certain cases involve relations which it is important to observe, either for purely analytical reasons, or for practically adapting the workings of the engine to their occurrence. (Ada 1842, Note D)

Pour éviter la confusion entre l'égalité $V_n = V_p + V_n$ (résultat de l'opération) et l'opération $V_n = V_p + V_n$ qui conduit au résultat, Ada introduit la notation ${}^{m+l}V_n = {}^{q}V_p + {}^{m}V_n$. pour conserver le caractère séquentiel de l'exécution.
Ada récupère ainsi *la notation d'usage pour les suites* pour signifier « la mise à jour » d'une colonne variable (en termes modernes). Le jeu d'indices d'états, approprié pour le travail sur les suites, se constitue alors en obstacle épistémologique à l'émergence de la notation d'affectation, indispensable à l'écriture d'une boucle. Pour justifier notre affirmation, appuyons nous sur un extrait du programme d'Ada destiné à calculer les nombres de Bernoulli (voir plus loin) donné dans le tableau 1 :

Number of operation	Nature of operation	Variables acted upon	Variables receiving results	Indication of change in the value on any Variable	Statement of results	Data		Working variables
						${}^{1}V_1$ **1**	${}^{1}V_3$ **4**	${}^{1}V_{10}$
1							n	
7	**-**	${}^{1}V_3$ - ${}^{1}V_1$	${}^{1}V_{10}$	$\begin{cases} {}^{1}V_3 = {}^{1}V_3 \\ {}^{1}V_1 = {}^{1}V_1 \end{cases}$	n – 1 (= 3)	1	1	**n - 1**
12	**-**	**${}^{1}V_{10}$ - ${}^{1}V_1$**	**${}^{2}V_{10}$**	$\begin{cases} {}^{1}V_{10} = {}^{2}V_{10} \\ {}^{1}V_1 = {}^{1}V_1 \end{cases}$	**n – 2 (= 2)**	**1**		**n – 2**
23	**-**	${}^{2}V_{10}$ - ${}^{1}V_1$	${}^{3}V_{10}$	$\begin{cases} {}^{2}V_{10} = {}^{3}V_{10} \\ {}^{1}V_1 = {}^{1}V_1 \end{cases}$	n – 3 (=1)	1		n – 3

Tableau 1. Extrait du programme de calcul des nombres de Bernoulli de Ada

Considérons la ligne 12 de ce tableau, les colonnes 3, 4 et 5 du tableau peuvent s'interpréter comme ${}^{1}V_{10}$ - ${}^{1}V_1$ = (n – 1) – 1 = n – 2 = ${}^{2}V_{10}$. On a une vraie égalité. Or cette égalité ne peut pas être transformée en une affectation V_{10} - $V_1 \rightarrow V_{10}$. De plus, la valeur initiale de V_{10} étant égale à n – 1, il est difficile de concevoir l'affectation : n – 1 – 1 $\rightarrow$ n – 1.
Cet obstacle épistémologique peut être interprété par le fait qu'Ada n'a pas à écrire un programme à une machine réelle. La présence réelle de mémoires dans une telle machine, en permettant une manipulation directe et effective des mémoires, permettrait-elle de surmonter cet obstacle ?
La notion de variable chez Ada n'est pas associée à une colonne variable dans la mémoire mais à un état précis (de cette colonne). Ainsi cette notion reste encore imprégnée de la notion de variable en mathématiques

I.3. Travail d'Ada pour permettre l'écriture d'un programme de calcul

Examinons maintenant le travail d'écriture d'Ada d'un programme de calcul des nombres de Bernoulli pour la machine Analytique (note G).
Elle part d'abord d'une formule bien connue à son époque :

$$\frac{x}{e^x - 1} = 1 - \frac{x}{2} + B_1 \frac{x^2}{1 \times 2} + B_3 \frac{x^4}{2 \times 3 \times 4} + B_5 \frac{x^6}{2 \times 3 \times 4 \times 5 \times 6} + \ldots$$

Il s'agit de calculer les coefficients B_1, B_3, etc. ou nombres de Bernoulli.
Pour permettre le calcul automatique, Ada transforme cette formule.
Elle démontre que :

$$B_{2n-1} = 2 \times \frac{1 \times 2 \times 3 \times \ldots \times 2n}{2\pi^{2n}} \times \left(1 + \frac{1}{2^{2n}} + \frac{1}{3^{2n}} + \ldots\right)$$ (avec n entier positif)

Elle en déduit la formule de récurrence :

$$-\frac{2n-1}{2(2n+1)} + B_1 \frac{2n}{2} + B_3 \frac{2n \times (2n-1) \times (2n-2)}{2 \times 3 \times 4} + \ldots + B_{2n-1} = 0$$

Ainsi le calcul des nombres Bernoulli se ramène au calcul suivant :

Calculer tous les nombres Bernoulli B_1, B_3, B_5 etc. connaissant la formule :

$$-\frac{2n-1}{2(2n+1)} + B_1 \frac{2n}{2} + B_3 \frac{2n \times (2n-1) \times (2n-2)}{2 \times 3 \times 4} + \ldots + B_{2n-1} = 0 \ (*)$$

Ainsi Ada transforme par un travail mathématique une formule existante contenant une infinité dénombrable de valeurs, en un processus itératif *fini* et peut ainsi utiliser la valeur n comme condition d'arrêt du calcul pour le compteur décrémenté. Elle justifie le processus de calcul des nombres B_{2n-1} obtenu comme suit :

This circumstance, combined with the fact (which we may easily perceive) that whatever *n* is, every term of (*) after the (*n*+1)th is = 0, and that the (*n*+1)th term itself is always $= B_{2n-1} \cdot \frac{1}{1} = B_{2n-1}$ enables us to find the value (either numerical or algebraic) of any *n*th Number of Bernoulli B_{2n-1}, *in terms of all the preceding ones*, if we but know the values of B_1, $B_3 \ldots B_{2n-3}$. […]
On attentively considering (*), we shall likewise perceive that we may derive from it the numerical value of *every* Number of Bernoulli in succession, from the very beginning, *ad infinitum*, by the following series of computations:
1st series. Let *n* = 1, and calculate (*) for this value of *n*. The result is B_1.
2nd series. Let *n* = 2. Calculate (*) for this value of *n*, substituting the value of B_1 just obtained. The result is B_3.
3rd series. Let *n* = 3. Calculate (*) for this value of *n*, substituting the values of B_1, B_3 before obtained. The result is B_5. And so on, to any extent. (Ada 1842, Note G)

En guise de conclusion

- L'écriture d'un programme et la description de son exécution virtuelle par la machine Analytique conduisent Ada à concevoir la notion de cycle (ou boucle dans les termes modernes). Cette notion lui permet de réduire le nombre de cartes et la longueur du programme, la longueur du programme étant égale au nombre d'opérations à exécuter. La seule boucle pouvant être présente est la boucle « Aller à… ».
- La machine de Babbage n'étant pas construite, Ada doit anticiper les changements des valeurs enregistrées sur les colonnes dans l'unité de mémoire (the Store) de la machine. Pour anticiper, elle invente la notion de « colonnes variables ». Cette notion lui sert à décrire le processus de changement de valeurs affectée aux colonnes durant l'exécution du calcul. Les exigences de cette description la conduisent à utiliser un système de double indice, emprunté à la notation mathématique pour les suites, pour indiquer les différents états des différentes « colonnes variables ». En termes modernes, nous parlerions de « mise à jour des variables ». Cette notation, marquée par le caractère séquentiel de l'exécution du programme, est cohérente avec l'explication mathématique qu'elle avance.
- Toutes les colonnes dans la mémoire sont appelées « variables », y compris les colonnes qui ne reçoivent que des valeurs constantes comme les valeurs d'entrée et les résultats finaux. Ce jeu d'indices d'état ainsi que l'absence d'une machine réelle s'oppose à l'émergence de la notion de variable informatique.
- Ada montre qu'un travail mathématique peut être nécessaire entre une solution mathématique connue et sa programmation. Par exemple, elle transforme une somme ayant

une infinité dénombrable de termes en une somme ayant un nombre fini de termes pour obtenir un algorithme de calcul itératif n-fini. Ce travail mathématique lui permet à la fois d'identifier un invariant de cycle et la formulation d'une condition d'arrêt (compteur décrémenté).

II. Boucle et variable pour la machine ordinateur de Von Neumann

Comme nous l'avons analysé chez Ada, Von Neumann dégage clairement deux aspects interdépendants dans la programmation : le travail mathématique sur la solution mathématique d'un problème et la programmation à une machine qui doit tenir compte des caractéristiques de la machine.

> L'un, purement mathématique, concerne le choix de l'expression appropriée; le second consiste à tirer le meilleur parti des caractères techniques de la machine, c'est-à-dire à faire exécuter de préférence les opérations qui sont accomplies le plus rapidement. Ces deux aspects ne sont pas séparables dans la pratique de la programmation. C'était la première fois que ce double aspect était souligné de manière si claire. (Ramunni 1989, p. 67)

Mais la conception d'un programme destiné à une machine en cours de construction peut de son côté faire évoluer la conception de cette machine. En 1945, Von Neumann et Goldstine écrivent les premières instructions destinées à la machine EDVAC (Electronic Discret Variable Computer), premier calculateur à programme enregistré, non encore complètement achevée, pour le problème de rangement d'une liste de nombres entiers. Certaines modifications apportées à EDVAC, en cours de construction (version I) résultent de leur travail d'écriture du programme.

> The program which now is in Dr. Goldstine's files is roughly 80 times faster, due to important improvement which Von Neumann considered shortly afterward. This second EDVAC design was apparently never defined in as much detail as the previous one, but a brief summary of its instructions codes appears in [5, p. 76][1] and we can deduce other properties by studying Von Neumann's program. (Knuth 1970, p. 252)

Comment peut-on communiquer avec une machine ordinateur de Von Neumann ?

II.1. Langage machine et possibilité de langage non machine

La machine ordinateur de Von Neumann comprend une unité de commande, une unité de calcul et de logique (contenus dans le processeur), une unité de mémoire qui contient plusieurs cases (cellules) et un système Entrée/Sortie.
Les caractéristiques principales du fonctionnement de l'ordinateur selon le principe de Von Neumann sont les suivantes :

+) L'ordinateur est mis en fonctionnement par un programme, écrit par l'homme et enregistré dans sa mémoire ;
+) Les programmes et les données sont enregistrés dans une unique mémoire ;
+) L'accès à la mémoire se fait par des adresses.

Ces caractéristiques rendent automatique le fonctionnement de l'ordinateur et augmentent considérablement les capacités et la puissance de l'ordinateur. Selon le principe de Von Neumann, le programme est considéré comme susceptible d'être lui-même l'objet d'un traitement avant ou pendant son exécution tout comme des données de calcul. Un programme, après avoir été exécuté peut devenir une instruction pour autre programme.
Dans les premières machines ordinateurs totalement électroniques (comme EDVAC ou EDSAC), un programme est écrit en un langage compris directement par la machine qui est traditionnellement appelé « langage machine ».

[1] Il s'agit de l'article de Eckert, J. Presper, Jr., et J-W. Mauchly "*Automatic high-speed computing: A progress report on the EDVAC*", Moor School 1945 (Originally classified "Confidential")

L'instruction 0001 110101 100111 111101 001101 à la machine EDVAC signifie (cf. http://www.miralab.unige.ch) :

Code opération +	Adresse du 1er nombre (53)	Adresse du 2e nombre (39)	Adresse du résultat (61)	Adresse de la prochaine instruction (13)
0001	110101	100111	111101	001101

Tableau 2. Signification de l'instruction 0001 110101 100111 111101 001101 à la machine EDVAC

Dans la machine EDSAC, la suite d'instructions : T 4095 ; A 100 ; A 101 ; T 101 additionnent deux nombres A et B placés dans deux cases de la mémoire aux positions 100, 101 puis placent la somme dans la case 101. (cf. Kuntzmann 1957).
La structuration sémantique des instructions est absente de ces programmes. Le codage des instructions d'affectation concerne l'aspect syntaxique du langage.
Notre analyse du traité de Kuntzmann (1957), atteste de la présence de la notion de boucle dans les programmes destinés à ces machines. Le seul type de boucle existante est « si…aller à… ». La notion de variable est implicitement assimilée à une case de la mémoire, mais elle ne fait pas l'objet d'une définition explicite.
Le langage à la machine EDSAC est plus « évolué » que celui à la machine EDVAC car ses instructions sont déjà composées à l'aide de codes mnémoniques et symboliques. (cf. Moreau 1987)
Les conditions, pour que les langages se libèrent des contraintes des codes machines et se rapprochent de la sémantique du langage naturel, sont maintenant réunies. En effet, une machine ordinateur selon le principe de Von Neumann rend possible l'enregistrement d'un programme pouvant traduire une instruction écrite en un langage « non machine » en codes compris directement par la machine. Pour le calcul de deux entiers par exemple, si le programme à la machine ordinateur entre l'instruction « A + B », le programme enregistré est appelé, il code « A + B » en langage machine (cf . tableau 2) pour exécution.
Désormais, l'amélioration des langages de programmation est possible et devient un enjeu crucial pour le développement de l'informatique. Et son histoire peut commencer. Nous tenterons plus loin d'en donner quelques jalons.

II.2. Notations d'affectation et de boucle d'un organigramme

Von Neumann et Goldstine distinguent deux étapes dans la codification d'un problème en langage machine, la première étape étant essentielle et donnant un rôle premier aux organigrammes : un organigramme permet de découper un problème en boîtes « s*équences autonomes et régulières de calculs* », puis d'interconnecter ces boîtes selon « l*a logique du problème* ». La deuxième phase consiste ensuite à coder les opérations contenues dans les boîtes.

> La manière la plus adéquate de programmer était d'établir un organigramme (*flow diagram*). Il s'agissait d'une représentation où le problème est découpé en séquences autonomes et régulières de calculs. Une fois cette division opérée, il fallait interconnecter, selon la logique du problème, les différentes parties considérées comme des boîtes où ce qui importe sont les données et d'entrée et de sortie. Von Neumann et Goldstine appellent cette première partie l'étape macroscopie de la codification. La phase suivante est la codification de chaque opération contenue dans les boîtes. Elle est qualifiée de statique ou microscopique. Dans cette phase, on numérote chaque étape élémentaire en employant d'abord une numérotation propre à chaque boîte et ensuite en renumérotant l'ensemble des opérations. (Ramunni 1989, p. 67)

L'organigramme permet donc de prendre en charge la sémantique et la structuration d'un programme, invisibles dans le langage machine : un organigramme est une représentation « boîtes, flèches » avec, à l'intérieur des boîtes, des instructions dans un langage de programmation libéré de la machine. La structuration du programme, utilisant boîte, flèches et langage, est particulièrement efficace pour visualiser une « s*équence autonome et régulière de*

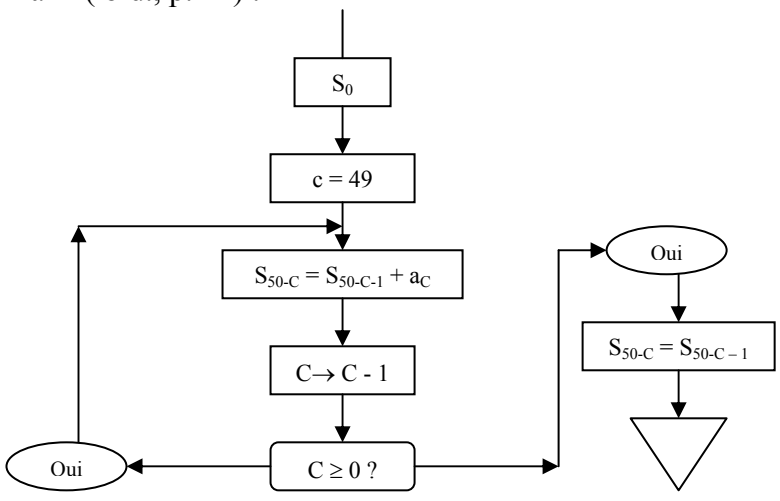
calculs » qui se répète, c'est-à-dire une boucle du programme. Seule la boucle « aller … à » (comme chez Ada) est présente dans les années 60, en l'absence de langage évolué.
C'est dans la boîte d'une boucle « aller … à » d'un organigramme que la notion d'affectation va apparaître.
Revenons au traité de Kuntzmann (1957) pour le montrer. Rappelons que dans ce traité les notions de variable et d'affectation d'une variable ne sont pas définies, mais cependant sont présentes en actes ou implicitement. Nous allons voir comment, en examinant un exercice de calcul itératif et ses solutions : programme machine commenté et organigramme.

Exemple : Somme de 50 termes avec le code d'exercice. Les a_i sont placés en 1000 + i. La somme s est en 200.

```
500   T      4095        } s_o = 0
501   T      200         }
502   A      49 P        }
503   U      198         } 49 en 198   (position de comptage)
504   A      199         } A 1001 en 199
505   T      506         } formation de l'instruction d'addition
506   A      1001 + n    }
507   A      200         } (1050 au début)
508   T      200         } s_n = s_n-1 + a_n
509   A      198         }
510   S      1 P         } n – 1
511   E      503
512   T      4095
513   A      200
514   Ordre  de sortie
515   Arrêt
```

(Kuntzmann 1957, p. 11)

L'enjeu didactique de l'exercice oblige Kuntzmann à contrecarrer l'indigence sémantique du programme machine en accompagnant ce programme de commentaires. Il reprend la notation indiciée des suites de l'énoncé dans les commentaires sur les instructions 506, 507 et 508. Il utilise aussi, pour structurer le programme machine, flèches et accolades comme commentaires non verbaux. Comparons le programme commenté à l'organigramme associé proposé par Kuntzmann (ibid., p. 11) :



Dans les boîtes de l'organigramme, apparaissent des instructions utilisant des expressions mathématiques analogues à celles des commentaires. Seules les instructions d'affectation notée « $c - 1 \rightarrow c$ » et de condition d'arrêt « $c \geq 0$? » se distinguent des commentaires. La volonté de représenter clairement la répétition d'une « s*équence autonome et régulière de calculs* » dans l'organigramme l'oblige à ces ajouts par rapport aux commentaires.
Kuntzmann donne quelques conseils pour l'écriture de la condition d'arrêt :

> Dans une machine utilisant des tests et non des comparaisons, on aura souvent intérêt à réaliser la sortie de la boucle en constatant la formation d'un zéro. Si l'on doit répéter 50 fois la même opération, *on commencera par mettre 50 dans une mémoire convenable* et après chaque opération, on enlèvera 1. *L'apparition d'un zéro (détecté par test)* sera le signe que le travail est terminé. (Kuntzmann 1957, fascicule 2, p. 10)

Ce qu'il appelle « mémoire convenable » est une variable informatique, en l'occurrence la variable « compteur décrémenté».

En guise de conclusion

Comme nous l'avons montré chez Ada, Von Neumann souligne l'importance du travail mathématique dans la programmation.
Notre analyse identifie une première condition pour l'émergence en actes de la notion de variable et de boucle qui est d'écrire un programme externe à une machine à mémoire effaçable (machine Analytique de Babbage et machine de Von Neumann). Cette identification rejoint les résultats de notre analyse de la co-genèse de la machine et du langage (chapitre B2).
Cependant les premiers langages en restant proches des caractéristiques technologiques de la machine opacifient la signification du programme et des notions qui y sont fondamentalement liées comme variable et boucle.
Cette opacité sémantique d'un langage machine est à la source de l'usage d'organigrammes et de l'avènement de langages de « haut niveau » proches de l'algorithme programmé et libérés des contraintes technologiques de la machine.
L'écriture de programmes de calculs itératifs dans des langages éloignés de la machine est une condition nécessaire à l'émergence d'une notation d'affectation dans une boucle qui permettra de rendre pleinement compte de la signification de la notion de variable en informatique.

III. Langages de programmations

Depuis les premiers programmes informatiques, les langages de programmations ont considérablement évolué. Un programme étant un ensemble d'instructions codées, nous illustrons cette évolution remarquable par trois exemples dans le tableau 3.

Programme pour la machine Analytique	Programme pour la machine EDVAC	Programme pour la machine ordinateur d'actuel

$^{1}V_{2} + {}^{0}V_{7}$
$^{1}V_{6} \div {}^{1}V_{7}$
$^{1}V_{21} \times {}^{3}V_{11}$
(Ada 1842, Note G)
	‘’1_{β}) $\bar{3}_{2} - \bar{3}_{2}$ | ℵ 0
	$2_{\beta})2\alpha \rightarrow C$
	
	(Von Neumann 1948, cité par Knuth 1970, p. 258)
		begin
		f : = 1
		for i : = 2 to n do
		f := f∗i
		end

Tableau 3. Évolution du codage des instructions : d'Ada au programme Pascal

Nous essayons d'abord de résumer cette évolution en termes de niveaux de langage puis nous examinerons parmi les langages évolués le style impératif.

III.1. Niveaux de langage

On distingue classiquement 3 types de langages : langage de bas niveau, langage assembleur et langage évolué (cf. Moreau 1987, p. 151).

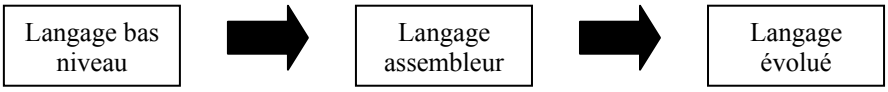


Figure 2. Les trois types de langage de programmation

Les tableaux de connexion des premières machines ordinateurs sont des langages de bas niveau, comme ceux des programmes communiqués au premier calculateur universel ENIAC (Electronic Numerical Integrator And Computer) :

> La programmation de ce calculateur s'effectue en recâblant entre eux, ses différents éléments. En effet, l'ENIAC n'est pas un ordinateur à programme enregistré. Il est divisé en 30 unités autonomes (dont 20 accumulateurs / additionneurs 10 digits, 1 multiplicateur et 1 "Master Programmer" capable de gérer les boucles). « Programmer » l'ENIAC consistait en fait à câbler toutes ces unités entre elles pour obtenir le résultat voulu. (Rossi 2003)

Un langage machine est très coûteux en apprentissage pour l'utilisateur de la machine, d'autant plus que chaque ordinateur a son langage :

> Pour pourvoir programmer en langage machine, l'utilisateur devait connaître tous les codes binaires représentant les instructions et savoir exactement dans quel emplacement mémoire, exprimé en binaire, se trouvaient les données. (Moreau 1987, p. 157)

Une nouvelle catégorie de langage voit donc le jour pour permettre à l'utilisateur de se détacher davantage des particularités des machines. Le langage assembleur (ou code mnémonique) apparaît avec l'un des premiers ordinateurs EDSAC. Ce type de langage comprend un ensemble d'instructions symboliques, d'adresse de mémoire etc. (cf. chapitre B2). Le langage assembleur est à la fois proche de la machine par l'utilisation du code-mnémonique et éloigné puisque les instructions utilisent des symboles. Ce langage assembleur présente l'inconvénient de rester encore trop proche de la machine :

> En effet la simplicité de leur syntaxe oblige le programmeur à penser son programme comme une suite d'instructions très liées à la structure de l'ordinateur, ce qui arrive à lui faire perdre de vue le problème à traiter. (Moreau 1987, p. 157)

L'ordinateur ne pouvant comprendre que le langage machine, un dispositif pour traduire un programme en langage assembleur permet de s'éloigner un peu plus des contraintes de la machine.
Deux types de traducteurs sont conçus: les interpréteurs et les compilateurs. Un interpréteur traduit et exécute les instructions du programme source (langage évolué) au fur et à mesure de leur lecture pour aboutir au résultat escompté. Un compilateur traduit l'ensemble du programme source pour produire un programme objet (langage machine). Nous schématisons ce processus de traduction dans la figure 3.

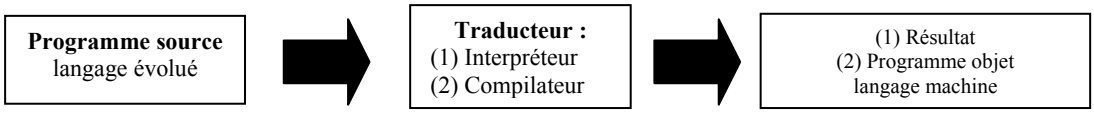


Figure 3. Processus de traduction d'un programme source

Le langage assembleur, lié aux caractéristiques de la machine ainsi qu'à l'architecture de l'ordinateur, rend difficile la communication entre l'homme et la machine ainsi que celle entre l'homme et l'homme lors de diffusion d'un programme.
Les langages dits de « haut niveau » sont une réponse à ces problèmes de communication.

> Ces langages permettent d'exprimer les algorithmes d'une manière lisible et aisément compréhensible par d'autres programmeurs. En outre, ces langages sont généralement « portables », c'est-à-dire qu'ils sont implémentés sur de nombreuses machines, de sorte que le programme peut être facilement « porté » (transféré) d'une machine à l'autre sans de profondes modifications. En ce sens ils sont dits « indépendants » de la machine. (Tucker 1986, p. 7)

Si un langage de bas niveau est contraint par le fonctionnement de la machine, un langage évolué est contraint par la structure de l'algorithme à programmer :

> Un langage de bas niveau est un langage orienté vers la machine, l'ordinateur ; il donne la possibilité d'écrire des instructions directement compréhensibles par la machine. Un langage de haut niveau est, au contraire, orienté vers le problème à résoudre : il permet de structurer – au sens algorithmique – le programme au plus près du problème. (Veigneau 1999)

Un langage évolué cherche donc à « structurer – au sens algorithmique – le programme au plus près du problème » pour reprendre les termes de Veigneau.

III.2. Langages impératifs et écriture de boucles

Rappelons que l'invention du compilateur (avec le langage Fortran), permise par le principe de programme enregistré de Von Neumann, a été la condition première pour l'évolution des langages :

> En 1957 […] plus personne ne mettait en doute l'efficacité des compilateurs, la communauté scientifique, passant alors d'un extrême à l'autre, prit l'habitude de définir in *abstracto* les langages de programmation, faisant hypothèse implicite que, s'il en était besoin, il serait toujours possible d'en rédiger un compilateur. (Moreau 1987, p. 170)

Depuis la naissance du langage Fortran (1954), des milliers de langages de programmation de haut niveau voit le jour. À titre indicatif, en 1967 il n'y avait que 120 langages (dont seulement 15 étaient vraiment utilisés), en 2003 plus de 2500 langages sont référenciés (cf. Hunault 2004).
Plusieurs critères sont possibles pour classer les langages évolués. Nous nous contenterons ici de nous placer dans l'un des quatre paradigmes de Hunault (2004)[1]. Un langage peut indiquer à la machine :

- « comment faire ? » : *langage impératif* (langages peu structurés comme Basic, Fortran / langage structurés par blocs comme Pascal, C / langages conçus comme langage objets Small Talk, C++, Java, Ada).
- « ce qu'on veut faire » : *langage déclaratif* (langage fonctionnels comme Lips / logiques comme Prolog).

Nous examinerons ici seulement le style impératif de la programmation.

Les premiers programmes à la machine ordinateur de Babbage ou de Von Neumann, que nous avons analysés dans la partie précédente, sont écrits dans le style impératif sous forme d'une succession d'ordre impérieux destinés à la machine. Les programmes en langage machine ou assembleur sont aussi de style impératif.

Langage impératif et machine
Moreau affirme que les langages de ce type « *modélisent plus ou moins les machines de Von Neumann* » :

> Tous ces langages modélisent plus ou moins les machines de Von Neumann. Certains auteurs comme Backus, les appellent d'ailleurs des langages de Von Neumann. (Moreau 1987, p. 151)

Un tel langage, tout en étant évolué, c'est-à-dire libéré des contraintes technologiques de la machine, garde des liens sémantiques avec la machine de Von Neumann, car « *la notion de mémoire est représentée par la donné abstraite qu'est une variable* ».

[1] Les autres paradigmes sont « l'objectif du langage » (scientifique, mathématique, gestion etc.), « le type de traitement » (compilation ou interprétation), « l'importance du langage » (script, interrogation, interfaces graphiques, interface web, quasi-universel ou ciblés) etc.

> Les langages majoritairement les plus utilisés actuellement sont ceux qui font partie de la catégorie des langages impératifs. Les ordinateurs étant des machines de Turing (améliorées par Von Neumann), la notion de mémoire est représentée par la donnée abstraite qu'est une variable, dans un langage impératif. D'autre part les machines de Turing sont séquentielles et les langages impératifs traitent les instructions séquentiellement. Ceci implique que les langages impératifs sont parfaitement bien adaptés à l'architecture de l'ordinateur ; ils sont donc plus « facilement » adaptables à la machine. (Di Scala 2004, p. 72)

Dans tout langage de programmation de style impératif, la mémoire de la machine est liée intrinsèquement à la notion de variable et à celle d'instruction qui indique, via un programme, la modification de l'état de la mémoire. Le style impératif (comme le langage C, Pascal ou Fortran) impose de donner du sens à la notion de variable via la mémoire de la machine.

> Le concept de variable correspond en fait à la notion de case de mémoire. Pour pourvoir lui donner un sens, il faut être conscient du fait que la machine est dotée d'une mémoire, composée de cases […] qui contiennent des valeurs susceptibles d'être lues et modifiées par des instructions. (Veigneau 1999, p. 239)

Langage impératif et structure algorithmique

En 1966, Böhm et Jacopini démontrent que tout programme, quelque soit sa complexité, peut se ramener à un programme composé de trois structures syntaxiques fondamentales : séquentialité, branchement, itération. Ce théorème connu sous le nom de « théorème de structure » fonde le paradigme de programmation structurée

Comment ces structures génèrent-elles des boucles pour les algorithmes itératifs ?

Pour répondre, nous donnons, dans le tableau 4, les quatre types de boucles connues écrites dans quatre langages impératifs évolués actuels.

Langages	Ada	Basic	C	Pascal
Boucle Si…aller à	**goto** label	**goto** n°	**goto** label	**goto** m
Boucle compteur	**for** i **in** intervalle **loop** Instructions **end loop** ;	**for** i = m **to** n Instructions **next**	**for** (i = m ; condition ; le pas p) Instructions	**for** i := m **to** n **do** Instructions
Boucle tant que	**while** condition **loop** Instructions **end loop** ;	**do** Instructions **while** condition **loop**	**while** condition Instructions	**while** condition **do** Instructions
Boucle répéter	**loop** Instructions **exit when** condition **endloop**	**do** Instructions **loop until** condition	**do** Instructions **while** condition	**repeat** Instructions **until** condition

Tableau 4. Différentes écritures de boucles dans des langages de programmation

Le problème de la boucle « Goto »

Depuis Ada, la boucle « Goto » est présente dans tous les langages de haut niveau comme on peut constater dans le tableau 4. Or cette boucle, attachée au fonctionnement de la machine, peut produire des programmes peu structurés. En 1968, prenant acte de ces risques, Dijkstra rédige dans les *Communications of the ACM* l'article fondateur "On the Go To Statement Considered Harmful", dans lequel il recommande l'abandon de l'usage de « Goto » dans les pratiques de programmation :

> Je crois de plus en plus que l'instruction **go to** devrait être abolie de tous les langages de « haut niveau » (c'est-à-dire tous les langages sauf, peut-être le langage machine). (Dijkstra 1968, p. 147) (Traduit par nous)

Pour Dijkstra, l'usage incontrôlé de la boucle « Goto » rend difficile la lecture du programme, et devient facultative du fait de l'existence d'autres structures d'itération comme « While » et « Repeat » :

> - Avec l'inclusion de clauses textuelles de répétition, la numérotation de ligne n'est plus nécessaire pour décrire la progression dynamique du processus ;
> - L'usage de manière incontrôlée de l'instruction go to amène à une conséquence immédiate : il est devenu terriblement dur de trouver l'ensemble significatif de coordination qui décrit la progression du processus ;
> - L'instruction go to est trop primitive ; elle appelle à faire un programme désordonné. (Dijkstra 1968, p. 148) (Traduit par nous)

De plus, un programme avec la boucle « Goto » peut conduire à des boucles enchevêtrées qui complexifient la validation du programme :

> Plus fondamentalement les tentatives de validation se heurtent à la détermination du « passé » d'une variable, qui devient très difficile, voir impossible à faire dans deux boucles qui se chevauchent. (Dijkstra 1972, cité dans Méjias 1985, p. 11)

Ainsi le paradigme d'une programmation structurée dans le style impératif repose sur trois types de boucle : « compteur », « tant…que » et « répéter…jusqu'à ». Nous regroupons ces différentes écritures de boucles en deux catégories selon que le nombre d'itérations est connu à l'avance ou non :
+) Boucle « compteur » : le nombre d'itération est déterminé à l'avance : « *répéter* n fois ... ». La variable compteur peut être décrémentée (comme dans le premier programme informatique écrit par Ada) ou incrémentée. Ce type de boucle simplifie fortement la syntaxe du langage de programmation puisque l'instruction de boucle se ramène à « répéter n fois ».
+) Boucle dont le nombre d'itérations n'est pas connue à l'avance. Dans ce cas il y a nécessité de formuler une condition logique d'arrêt pour sortir de la boucle. Cette condition peut être placée juste après l'entrée de la boucle « *vérifier condition et répéter*... » ou juste avant la sortie de la boucle « *répéter*…*vérifier condition* ».

Dans la formulation d'une boucle, nous avons mis en évidence le risque de confusion entre l'égalité mathématique et l'opération d'affectation, qui elle-même peut entraîner une confusion entre variable informatique et variable mathématique.
Comment les différents langages répondent-ils à cette difficulté ? Le tableau 5 montre, selon le langage, l'usage de l'égalité dans les notations d'affectation et dans l'instruction conditionnelle qui compare l'identité de deux nombres :

Langages	Ada	C	Pascal	Basic
Affectation	a := a + b	a = a + b	a:= a + b	a = a + b
Condition	**If** a = b	**If** a = = b	**If** a = b	**If** a = b

Tableau 5. L'usage de l'égalité dans l'affectation et dans les instructions conditionnelles selon les langages

Les langages C et Basic utilisent tous les deux le signe « = » pour l'affectation. Le langage C distingue bien l'égalité mathématique de l'opération d'affectation par des signes différents mais curieusement, il attribue « = = » pour noter l'égalité ! Le langage Basic renforce la difficulté en utilisant le même signe « = » pour l'affectation et l'égalité.

Résumé et conclusion

Les premiers langages en restant proches des caractéristiques de la machine rendent opaques la signification du programme et des notions qui y sont fondamentalement liées comme variable et boucle. Cette opacité est à la source de l'usage d'organigrammes et de l'avènement de langages évolués proches de l'algorithme programmé et libres des contraintes

technologique de la machine. De tels langages, tout en étant évolués, c'est-à-dire libérés des contraintes technologiques de la machine, gardent des liens sémantiques avec la machine de Von Neumann, car « *la notion de mémoire est représentée par la donnée abstraite qu'est une variable* ».

Mais si un langage de bas niveau est contraint par le fonctionnement de la machine, un langage évolué est contraint par la structure de l'algorithme à programmer :

> Un langage de bas niveau est un langage orienté vers la machine, l'ordinateur ; il donne la possibilité d'écrire des instructions directement compréhensibles par la machine. Un langage de haut niveau est, au contraire, orienté vers le problème à résoudre : il permet de structurer – au sens algorithmique – le programme au plus près du problème. (Veigneau 1999)

Un langage évolué cherche donc à « structurer – au sens algorithmique – le programme au plus près du problème » pour reprendre les termes de Veigneau.
Dans tout langage de programmation de style impératif, la mémoire de la machine est liée intrinsèquement à la notion de variable et à celle d'instruction qui indique, via un programme, la modification de l'état de la mémoire.

Ainsi, *l'écriture de programmes de calcul itératif en ces langages éloignés de la machine est une condition nécessaire à l'émergence la notation d'affectation dans une boucle* qui permet de rendre pleinement compte de la signification de la notion de variable en informatique.

Nous avons aussi mis en évidence le risque de confusion entre l'égalité mathématique et l'opération d'affectation, qui elle-même peut entraîner une confusion entre variable informatique et variable mathématique. Certains langages évolués du style impératif favorisent ce risque en utilisant la notation d'égalité en mathématique « = » pour l'opération d'affectation de variable (langage Basic, C ou Fortran). En particulier, Basic utilise cette notation pour la comparaison de deux variables. *Une deuxième condition sur les langages pour éviter ce risque de confusion est la distinction sémiotique entre l'égalité et l'affectation*.

Selon que le nombre d'itérations est connu à l'avance ou non, la répétition peut être structurée en langage impératif selon deux types principaux de boucles : boucle compteur et boucles conditionnelles. L'avantage de la boucle compteur « *répéter* n fois ... » est d'être simple sur le plan syntaxique, même si la construction de la boucle n'a pas nécessairement lieu pas sur le plan sémantique.

Conclusion de la partie B

Notre enquête épistémologique a permis d'identifier dans l'histoire de l'informatique et de son enseignement un certain nombre de conditions favorables à la genèse des notions de variable et de boucle.
Ces conditions repérées nous conduisent à faire des choix pour la conception d'une ingénierie didactique d'introduction non seulement des objets de base de l'algorithmique et de la programmation mais aussi des notions de machine et de langage à la machine. Ces choix et ces conditions énoncés sous la forme d'hypothèses seront mis à l'épreuve de la réalisation de l'ingénierie didactique et de son analyse.

Choix d'une stratégie d'enseignement de notions d'algorithmique et de programmation

L'analyse des traités (chapitre B1) met en évidence deux stratégies d'enseignement :

- présence d'une machine de référence, réelle ou fictive. Ces machines fictives peuvent être virtuelle (c'est-à-dire non construite comme la machine de Babagge, 1842), ou idéale (c'est-à-dire non construite mais représentante d'une classe de machines réelles comme la machine MIX de Knuth, 1968). L'enseignement de l'algorithmique est alors lié à la compréhension de l'architecture de la machine et à la conception d'un langage approprié à cette machine.

- absence de machine de référence : on vise un langage de programmation représentatif d'une classe de langages existants (comme le langage SPARKS de Horowitz, 1978), *ce qui suppose la connaissance préalable à l'enseignement de l'algorithmique d'un langage de programmation*.

L'absence institutionnelle de tout langage de programmation au niveau secondaire (aussi bien en France qu'au Viêt-nam (chapitres A2, A3)) et l'analyse faite de ces deux stratégies d'enseignement nous conduisent à formuler un choix fondamental.

Hypothèse 1 (chapitres A2, A3, B1)
En l'absence de langage de programmation, un enseignement d'algorithmique et de programmation doit prendre comme enjeu didactique l'architecture d'une machine et ses relations avec un langage de programmation (comme dans la première stratégie d'enseignement).

Choix d'un message à une machine à mémoire effaçable

Du point de vue de l'architecture de la machine, les trois étapes « Machine Arithmétique », « Machine Analytique » et « Machine Ordinateur » sont marquées d'une part par l'accroissement des possibilités de stockage et d'autre part par la propriété fondamentale d'effaçabilité de la mémoire qui, conjointement, accroissent les possibilités de calcul de la machine.
L'évolution de la mémoire vers le stockage et l'effaçabilité de résultats intermédiaires (machine de Babbage) est l'un des facteurs qui rend possible l'exécution mécanique de calculs répétitifs, c'est-à-dire d'un algorithme itératif de calcul.
L'idée qu'un programme dit enregistré et des données puissent partager une mémoire commune (machine de Von Neumann) permet de traiter un programme comme des données de calculs. Un programme, une fois exécuté, peut devenir une instruction pour un autre programme.
L'exécution du programme d'un algorithme itératif, avec boucle, nécessite aussi de coordonner un calcul (unité de calcul) et les résultats intermédiaires (unité de mémoire). Un

second facteur, celui de l'existence d'une unité de commande comme dans les machines de Babbage et de Von Neumann, permet de déléguer à la machine la décision de commencer, d'arrêter un calcul ou de l'articuler à d'autres. Cela a pour conséquence fondamentale de permettre l'écriture d'un programme à une machine.
Nous schématisons cette évolution dans la figure 1 ci-après :

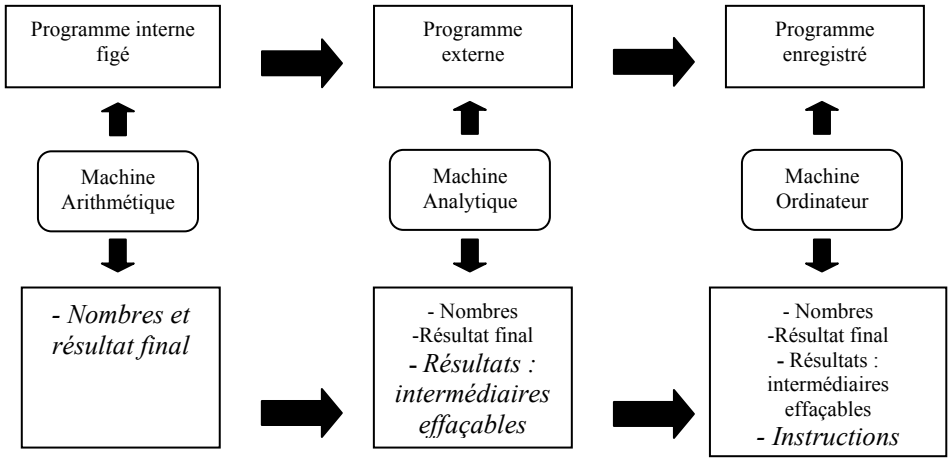


Figure 1. Évolution des mémoires et de la programmation selon le type de machines

Écriture d'un message à une machine ordinateur de Von Neumann

La machine ordinateur de Von Neumann avec le concept de programme enregistré a permis la traduction de tout langage proche sémantiquement du langage naturel en un langage codé exécutable par la machine. Rappelons qu'un tel langage est un langage évolué.
Il en a résulté, comme nous l'avons montré dans le chapitre B3, un éloignement progressif des langages du langage machine, pour se rapprocher de la signification de l'algorithme à programmer pris en charge par trois structures fondamentales : séquentialité, branchement et itération.

Hypothèse 2 (chapitre B2 et B3)
L'écriture d'un message à une machine ordinateur de Von Neumann pour obtenir l'exécution d'un algorithme itératif de calcul est une condition favorable à l'émergence en actes d'éléments d'un langage évolué : séquentialité, branchement et itération.

Signification de la notion de variable informatique

Un langage codé (comme le langage machine) restant trop proche des caractéristiques technologiques de la machine masque, dans l'écriture du programme, la signification de l'algorithme programmé. En langage évolué, l'écriture d'une boucle pour un algorithme itératif, nécessite d'affecter successivement des valeurs à une mémoire effaçable. La notion d'affectation permet donc de rendre pleinement compte de la signification de la notion de variable informatique : une variable informatique est une mémoire effaçable désignée pour recevoir des valeurs successives.

Hypothèse 3 (chapitre B3)
L'écriture d'un programme de calculs répétitifs (algorithme itératif) *en un langage proche du fonctionnement de la machine* est une condition favorable à l'émergence de la notion de variable comme mémoire effaçable.
En particulier, la distinction sémiotique entre l'égalité et l'affectation peut renforcer la distinction entre l'égalité mathématique et l'opération d'affectation.

Un travail mathématique nécessaire

L'exemple historique du premier programme informatique écrit par Ada montre qu'un travail mathématique est nécessaire entre une solution mathématique connue d'un problème et sa programmation.
Ce travail mathématique de nature algorithmique doit permettre à la fois d'identifier un invariant de boucle et la formulation d'une condition d'arrêt.

Hypothèse 4 (chapitres B2 et B3)

La formulation de la condition d'arrêt et de l'invariant de la répétition, nécessaire à l'écriture d'un message à une machine à mémoire effaçable, oblige à un travail réflexif sur les objets mathématiques présents dans la solution mathématique d'un problème.

Dans la partie suivante, nous formulons une situation fondamentale de l'algorithmique et de la programmation qui articule un problème mathématique et un problème informatique et nous l'étudions (chapitre C1) ; cette situation fondamentale exige l'écriture d'un message à une machine. C'est pour cela que dans une deuxième partie nous questionnons l'architecture des calculatrices présentes dans EMS notamment en ce qui concerne les mémoires et leur effaçabilité.

Partie C

Algorithmique et programmation

Choix macro-didactiques pour une ingénierie didactique

Chapitre C1
Une situation fondamentale de l'algorithmique et de la programmation
Tabulation d'une fonction numérique par une machine

Introduction

Dans la partie B, nous avons montré le rôle crucial joué par le calcul numérique et en particulier par la tabulation des fonctions numériques dans la conception des machines et dans leur évolution conjointe avec celle des langages.
La situation de tabulation d'une fonction numérique par une machine est donc pour nous une situation fondamentale de l'algorithmique et de la programmation. Elle articule deux problèmes :

• Un problème mathématique de tabulation décimale d'une fonction numérique

Calculer les images décimales, par une fonction **f**, de **m** valeurs décimales de la variable réelle x : x_0, $x_1 \ldots x_{m-1}$ espacées d'un pas **p** décimal fixe.

La figure 1 ci-après schématise les problèmes mathématiques particuliers générés par des valeurs données aux variables (en gras dans l'énoncé) du problème mathématique énoncé. Ces deux grandes classes donnent lieu à des algorithmes connus de résolution, listés pour certains dans la figure 1. Parmi ces algorithmes, solutions du problème mathématique fondamental, certains ont pu être produit par un travail mathématique sur des solutions existantes pour tenir compte des particularités d'une machine qui doit les exécuter, comme le programme d'Ada pour la machine analytique de Babbage (chapitre B3).

Ces problèmes de tabulation peuvent concerner :
1. des fonctions dont la formule analytique est inconnue mais dont on connaît les images pour certaines valeurs de la variable ;
2. des fonctions dont la formule analytique est connue comme des fonctions polynomiales ou des fonctions transcendantes.

Nous explorons successivement ces deux classes de problèmes. Cependant nous restreignons, en le justifiant, la deuxième classe aux fonctions polynomiales.
La résolution du problème de tabulation exige la répétition de calculs identiques comme l'attestent les algorithmes de calcul présentés dans la suite de ce chapitre.
L'amélioration de la fiabilité des résultats des calculs et la réduction du coût de la répétition de ces calculs ont conduit historiquement à concevoir des machines et à écrire des programmes adaptés aux caractéristiques de cette machine (partie B).
C'est pour cela qu'une situation fondamentale de l'algorithmique et la programmation articule un problème de tabulation à un problème informatique.

• Un problème informatique

Il s'agit d'écrire un programme informatique pour l'exécution d'une solution du problème mathématique précédent par une machine à mémoires effaçables.

Les algorithmes, solutions du problème mathématique, sont comparés du point de vue du coût en nombre d'opérations à exécuter (temps de calcul), mais aussi du point de vue du coût en mémoires utilisées et du coût de l'écriture du programme. Nous considérons que les notions de variable et de boucle, nécessitant une machine à mémoire effaçable, sont une réponse au problème des coûts de l'écriture et en mémoires utilisées.

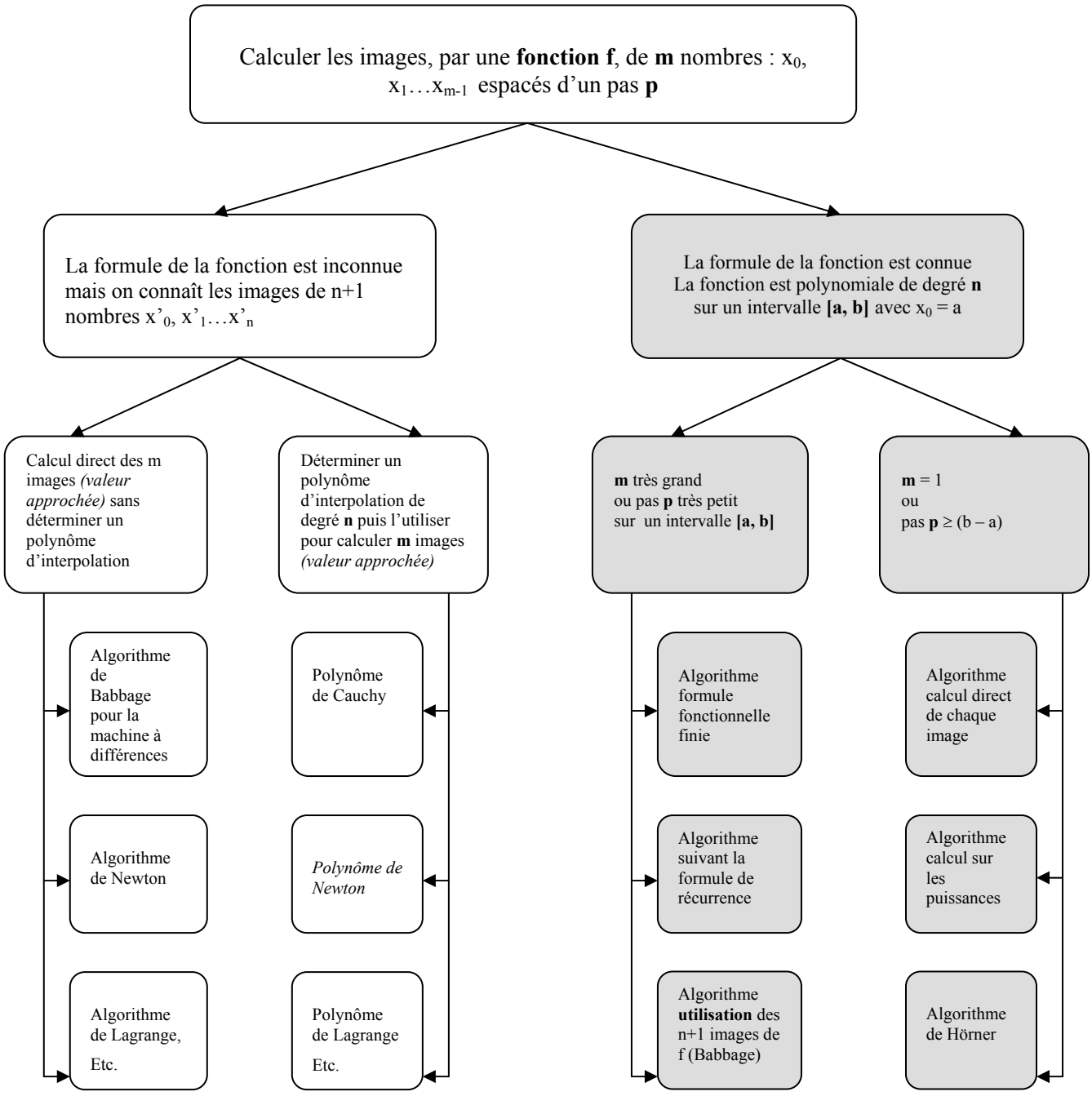


Figure 1. Génération de problèmes à partir de valeurs des variables du problème général

I. Le problème de la tabulation d'une fonction numérique dont la formule est inconnue

Nous présentons d'abord des algorithmes de calculs solutions de ce problème et quelques résultats théoriques mathématiques qui les fondent, puis nous comparons ces algorithmes du

point de vue du coût de l'écriture dans un programme informatique pour leur exécution par une machine à mémoires effaçables (problème informatique).

Ces algorithmes de calcul peuvent suivre deux schémas de calcul :

Schéma 1

• Déterminer une fonction polynôme P de degré n grâce à n + 1 images par la fonction de n + 1 nombres.
P est appelé polynôme d'interpolation : polynôme d'interpolation de Newton, de Lagrange, etc.
• Puis utiliser P pour calculer les images par P des m nombres. Ces m images sont des valeurs approchées des images par f des m nombres.

Schéma 2

• Calculer directement m valeurs à partir des valeurs données (sans devoir déterminer un polynôme d'interpolation) comme l'algorithme de Newton ou l'algorithme utilisé par Babbage.

La notion de différence finie est utilisée dans le deuxième schéma de calcul mais aussi pour déterminer certains polynômes d'interpolation comme celui de Newton. C'est pour cette raison que nous commençons par exposer les fondements théoriques de cette notion.

I.1. Différences finies et dérivées successives

La notion de différence finie est née au 17ième siècle de la volonté des mathématiciens de l'époque de rendre plus précis les calculs nécessaires à l'établissement de tables numériques.

> Dans un premier temps, pour calculer ou mettre en œuvre des tables numériques, les mathématiciens ont utilisé l'interpolation linéaire. C'est en tout cas ce que suggère Ptolémée à son lecteur pour connaître les valeurs des cordes d'arc de minutes en minutes. L'interpolation linéaire n'est cependant pas toujours suffisante pour obtenir de bonnes sous-tabulations, c'est-à-dire des valeurs intermédiaires assez précises. C'est ainsi que les mathématiciens ont été conduits à élaborer un nouveau calcul à l'aide des différences finies […] (Chabert et al., 1994, p. 369)

Nous nous appuyons sur Chabert (1994) pour faire une brève présentation de cette notion. Nous utilisons les notations actuelles.
Supposons que l'on connaisse les valeurs $f(x_0),\ldots,f(x_n)$ d'une fonction f. On associe à cette suite de valeurs $f(x_k)$ la suite de leurs différences finies d'ordre 1 :
$\Delta f(x_0) = f(x_1) - f(x_0)$, $\Delta f(x_1) = f(x_2) - f(x_1),\ldots,\Delta f(x_{n-1}) = f(x_n) - f(x_{n-1})$;
puis d'ordre 2 :
$\Delta^2 f(x_0) = \Delta(\Delta f(x_0)) = \Delta f(x_0) - \Delta f(x_1) = (f(x_0) - f(x_1)) - (f(x_1) - f(x_2))$ et ainsi de suite.

Plus généralement, la différence finie d'ordre r est calculée par :

$$\Delta^r(f(x_k)) = \sum_{0 \le s \le r} (-1)^{r-s} \binom{r}{s} f(x_{k+s}) \qquad (1)$$

De Morgan (1842) définit les « différences finies divisées ». Celle d'ordre 1, par exemple, est ce que nous appelons actuellement le taux de variation :

$$\underline{\Delta}(f(x_k)) = \frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} ;$$

Si les valeurs x_i sont espacées régulièrement d'un pas p, alors

$$\underline{\Delta}(f(x_k)) = \frac{\Delta(f(x_k))}{p}.$$

On peut montrer par récurrence la relation existant entre les différences finies et les différences finies divisées :

$$\underline{\Delta}^{i}\left(f\left(x_k\right)\right)=\frac{\Delta^{i}\left(f\left(x_k\right)\right)}{i!\,p^{i}} \text{ où p est le pas} \qquad (2)$$

Le théorème d'analyse suivant fait le lien entre la dérivée nième d'une fonction (dérivable jusqu'à l'ordre n) et la différence finie d'ordre n.

> Soit une fonction f dérivable jusqu'à l'ordre n, alors : $\Delta^n f(x_0) = p^n f^{(n)}(\xi)$ où p est le pas, $\xi \in [x_0, x_n]$ (Kuntzmann 1969, p.116)

Lorsque f est une fonction polynomiale, alors la dérivée d'ordre n est égale à $a_n n!$. Ainsi à partir du théorème précédent, nous obtenons le corollaire suivant :

> Soit f une fonction polynomiale de degré n alors $\Delta^n f = a_n p_n (n!)$ où p est le pas, a_n est le coefficient de x_n.

La différence finie d'ordre n d'une fonction polynomiale f est un nombre constant (3).
L'algorithme utilisé par Babbage pour sa machine à différence finie (chapitre B2) repose sur ce résultat mathématique. Nous y revenons dans le paragraphe suivant (I.2.2).

I.2. Calcul sans déterminer de polynôme d'interpolation

La machine à différences de Babbage (chapitre B2) peut exécuter un algorithme de calcul utilisant les différences finies.
Prenons l'exemple suivant (fonction approchée implicitement par une fonction polynomiale de degré 2) :
Connaissant les images par une fonction f de trois valeurs de variable espacées régulièrement d'un pas p, le problème est de calculer la quatrième valeur, puis la cinquième, ainsi de suite.
L'algorithme utilisé par Babbage est alors :

Valeurs de la variable x	**Images par la fonction f**	**Différence première Δf**	**Différence deuxième Δ^2f**
$x_0 = -2$	**$f(x_0) = 5$**	**- 3**	**2**
$x_1 = -1$	**$f(x_1) = 2$**	**-1**	*2*
$x_2 = 0$	**$f(x_2) = 1$**	*1*	
$x_3 = 1$	$f(x_3) = 2$		
…	…		

+ (entre -1 et 1) ; + (entre 1 et 2)

Le processus de calcul de la machine à différence sur cet exemple est le suivant :

- Calculer les deux premières différences finies d'ordre 1 de cette fonction :
  $\Delta f(x_0) = f(x_1) - f(x_0) = -3$, $\Delta f(x_1) = f(x_2) - f(x_1) = -1$.
- Calculer la première différence finie d'ordre 2 de cette fonction :
  $\Delta^2 f(x_0) = \Delta(\Delta f(x_0)) = \Delta f(x_0) - \Delta f(x_1) = 2$.
- Si on considère que f est une fonction polynomiale, alors toutes les différences d'ordre 2 sont égales à 2 (résultat (3)).
- Ajouter 2 à la dernière différence d'ordre 1 déjà calculer : 2 + (- 1) = 1 puis ajouter le résultat obtenu à la dernière image de la fonction déjà calculée : 1 + 1 = 2. Ce résultat est l'image de x_3 par la fonction f.

Et on réitère ce processus de calcul le nombre de fois souhaité. Par exemple, si on veut calculer la valeur $x_k = x_0 + k \times p$ (k entier ≥ 4), il faut répéter ce processus k – 3 fois.

I.2.1. Comment la machine à différences finies de Babbage effectue-t-elle cet algorithme de calcul ?

Soit à calculer $f(x_{n+1})$.

Sont déjà enregistrés sur les mémoires colonnes de la machine :
- (n + 1) images par la fonction f des nombres x_0, $x_1 \ldots x_n$ (espacées d'un pas p),
- n différences finies d'ordre 1,
- n – 1 différences finies d'ordre 2
…
- 1 différence finie d'ordre n

Il y a donc $\frac{(n+2)\times(n+1)}{2}$ valeurs à enregistrer.

> Les données initiales d'un calcul sont entrées, après le déblocage des roues chiffrées, par une rotation manuelle jusqu'à la valeur décimale adéquate. (Swade 1993, p. 80)

Puis les roues chiffrées sont de nouveau tournées manuellement pour le calcul d'une image de la fonction, par exemple $f(x_{n+1})$ où $x_{n+1} = x_n + p$.
La rotation de la manivelle, activée par un opérateur humain, produit donc la valeur $f(x_{n+1})$ et génère les valeurs $\Delta^1 f(x_{n+1})$, $\Delta^2 f(x_{n+1})$, $\ldots\Delta^{n-1} f(x_{n+1})$, $\Delta^n f(x_{n+1})$ ($\Delta^n f(x_{n+1})$: nombre constant) qui sont « enregistrées » sur les colonnes variables. Ces nouvelles valeurs sont de résultats intermédiaires pour le calcul de l'image de x_{n+2}. Et ainsi de suite.

> Chaque tour de manivelle produit une valeur à 30 chiffres dans la table des différences et prépare automatiquement la machine à générer le nombre suivant. (Swade 1993, p. 80)

Les colonnes variables sur lesquelles les résultats intermédiaires sont enregistrés sont de même nature que les lucarnes dans la « Pascaline », c'est-à-dire la machine arithmétique de Pascal (chapitre B2). Si on se réfère à la calculatrice, cette mémoire est de même nature que la mémoire du dernier résultat « Ans » dans la calculatrice (voir chapitre suivant).
Babbage a donc fait un travail mathématique pour produire un algorithme adapté aux caractéristiques de sa machine, en particulier sans mémoire effaçable. Son algorithme exige seulement l'opération arithmétique la plus simple, l'addition et la mémoire « du dernier résultat » (colonne de la machine).

Quel est le coût de ce calcul en nombre d'opérations de la machine et en opération humaine ?
Cet algorithme nécessite n additions pour chaque calcul d'une image. À chaque image produite, l'opérateur humain doit tourner la manivelle pour le calcul de l'image suivante. Pour calculer l'image approchée de $x_0 + k \times p$ par f, il faut donc $n \times (k - n)$ additions, chaque n-additions étant séparée par un tour de manivelle de l'opérateur.

I.2.2. Comment une machine à mémoire effaçable effectue-t-elle cet algorithme de calcul ?

Nous décrivons l'algorithme de calcul de Babbage en langage algorithmique pour calculer des images d'une fonction approchée implicitement par une fonction polynomiale de degré 2 (exemple précédent).

```
Début
Lire le rang k de x
- 1 → a            {initialiser la 2e différence finie d'ordre 1}
  1 → b            {initialiser la 3e image de la fonction}
Pour i de 1 à k – 3
  2 + a → a
  b + a → b
Fin pour
Afficher b         {b est l'image de la fonction correspondante à x = x0 + k × p}
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?
Les calculs de chaque image d'une fonction, approchée implicitement par une fonction polynomiale, nécessitent n additions. Le calcul des m images des m nombres $x_{n+1},\ldots, x_{n+m}$ nécessitent $n \times (m - n)$ additions. Il faut aussi définir n variables informatiques, c'est-à-dire n emplacements dans l'unité de mémoire.

I.3. Détermination d'un polynôme d'interpolation

Le fondement théorique du problème traité est le suivant :

> Connaissant les n + 1 images d'une fonction f en n + 1 valeurs x_i de la variable x, on peut l'approcher par une fonction polynomiale de degré n. Ce polynôme P est appelé *polynôme d'interpolation* de la fonction f en x_i.

Le calcul approché d'une image de la fonction f en x est ramené au calcul de la valeur du polynôme en x.

> Il est possible de trouver, pour une fonction continue dans intervalle [a ; b] des approximations polynomiales aussi bonne que l'on veut (Kuntzmann 1969, p. 110)

On ne peut donc pas parler d'interpolation sans évoquer l'idée d'approximation. L'approximation d'une fonction et l'interpolation d'une fonction sont deux aspects distincts. La différence réside essentiellement dans la connaissance ou non de la formule de la fonction :

> L'interpolation d'une fonction sur un intervalle correspond au calcul approché de la valeur d'une fonction en un point à partir de la donnée d'autres valeurs, cette fonction n'étant pas nécessairement entièrement déterminée ; l'approximation d'une fonction traduit une idée plus globale, il s'agit de remplacer une fonction mal connue ou d'expression compliquée par une fonction connue ou plus simple qui soit proche de la fonction initiale sur tout un intervalle et non pas en un seul point. Le terme « proche » reste bien sûr à préciser dans la mesure où différentes « proximités » sont possibles. (Chabert et al. 1994, p. 356)

Nous donnons l'exemple du polynôme d'interpolation de Newton.

> Nous avons :
> $P = f(x_0) = \Delta^1 f(x_0)(x - x_0) + \Delta^2 f(x_0)(x - x_0)(x - x_1) + \Delta^3 f(x_0)(x - x_0)(x - x_1)(x - x_2) + \ldots$ (Chabert et al. 1994, p. 377)

En utilisant la relation (2) entre les différences finies et les différences finies divisées, nous obtenons :

$$P(x) = f(x_0) + \frac{x - x_0}{1!\,p}\Delta^1 f(x_0) + \frac{(x - x_0)(x - x_1)}{2!\,p^2}\Delta^2 f(x_0) + \ldots + \frac{(x - x_0)(x - x_1)\ldots(x - x_{n-1})}{n!\,p^n}\Delta^n f(x_0) + \ldots$$ où

> p est le pas

Le polynôme d'interpolation de degré n de Newton est obtenu en tronquant au rang n le second membre.
Exemple : Sachant les valeurs f(-2) = 5 ; f(-1) = 2 ; f(0) = 1 d'une fonction, nous pouvons calculer d'autres valeur en déterminant d'abord le polynôme d'interpolation de Newton de cette fonction comme suit : $\Delta^1 f(x_0) = -3$, $\Delta^2 f(x_0) = 2$.
Ainsi : $P_2(x) = f(x_0) + \frac{x + 2}{1!\,p}\Delta^1 f(x_0) + \frac{(x + 2)(x + 1)}{2!\,p^2}\Delta^2 f(x_0)$ = 5 - 3(x + 2) + (x + 2)(x + 1).
A partir de ce polynôme on peut alors calculer d'autres valeurs de la fonction.

Quel est le coût de ce calcul comparé à l'algorithme de Babbage pour la machine à différence ?

Pour calculer l'image approchée de $x_0 + k \times p$ par f, il faut : $\frac{n(n-1)}{2}$ additions, n divisions et $n^2 + 2n + 1$ multiplications. Ce calcul est plus coûteux que l'algorithme de Babbage. De plus, il est plus difficile de mécaniser la multiplication et la division que l'addition. Cela peut

expliquer les décisions de Babbage de ne pas choisir cet algorithme connu à l'époque et d'en concevoir un autre.

II. Calcul en connaissant la formule de la fonction

Nous nous limitons au cas de la fonction polynomiale qui joue un rôle central pour le problème de tabulation comme nous venons de le voir. Sous certaines conditions (connaître les images d'une fonction f en (n + 1) nombres par exemple) on peut approcher n'importe quelle fonction réelle par une fonction polynomiale de degré n, appelé polynôme d'interpolation.

> L'importance de ce problème est due en une partie au fait que des fonctions classiques comme sin x, cos x, ex, etc. sont souvent calculées par des procédures qui sont liées à l'évaluation de certains polynômes ; comme ces polynômes sont très souvent évalués il est souhaitable de trouver la méthode la plus rapide à faire ce calcul. (Knuth 1971, p. 432)

Nous allons examiner différents algorithmes de calcul des m images par une fonction polynomiale f de m nombres espacées d'un pas p de la variable x appartenant à l'intervalle [a ; b].
Nous considérons deux cas :

- m = 1, c'est-à-dire calculer l'image par une fonction polynomiale d'un seul nombre ;
- m > 1, c'est-à-dire calculer l'image par une fonction polynomiale de plusieurs nombres.

II.1. Calcul de l'image d'un nombre v par une fonction polynomiale connue

Soit le polynôme : $f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_o$, $(a_n \neq 0)$
Le coût du calcul à ce problème s'avère crucial. Nous allons présenter quelques algorithmes, comparer leur coût en calculs et examiner les possibilités de réduire ces coûts.

II.1.1. Algorithme de calcul direct

Nous appelons algorithme direct le processus de calcul consistant à :

- calculer chaque termes $a_n x^n$, $a_{n-1}x^{n-1}$, $\ldots a_1 x$, a_o en remplaçant x par v
- additionner les résultats.

C'est la solution mathématique la plus simple.
Pour l'organisation de données, il faut :

1. mémoriser tous les coefficients de la fonction polynomiale dans un premier tableau à une dimension a[1..n][1].
2. mémoriser tous les résultats intermédiaires dans un deuxième tableau à une dimension b[1..n] ;

Nous décrivons ce processus de calcul comme un programme en langage algorithmique à une machine à mémoire effaçable

```
Début
Initialiser a[1..n] ; v
Pour i de 1 à n
        a[i]*v^i → b[i]              {calculer chaque terme a[i]*v^i est mémorisé dans b[i]}
Pour i de 1 à n
        calculer la somme des b[i]   {calculer la somme de tous les termes}
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?

Les calculs de chaque terme nécessitent : $n + (n - 1) + \ldots + 1 = \frac{n(n+1)}{2}$ multiplications. Le calcul de la somme des résultats nécessite n additions.

[1] Nous adoptons la notation du langage algorithmique.

En termes de mémoire, il faut deux tableaux a[1..n] et b[1..n] de dimension 1.

II.1.2. Algorithme de calcul sur les puissances

L'algorithme consiste à :

- former une table [1..n] pour contenir les valeurs de x, x^2, x^3…x^{n-1}, x^n quand on remplace x par v. Un terme est calculé en multipliant le terme qui le précède par v.
- multiplier chaque terme par son coefficient
- additionner tous les produits (cf. Knuth 1971, pp. 422-423).

Pour l'organisation de données il faut :

- mémoriser les coefficients de la fonction polynomiale dans un premier tableau à une dimension a[1..n] ;
- mémoriser tous les résultats intermédiaires dans un deuxième tableau à une dimension b[1..n] ;

Nous décrivons ce processus de calcul comme un programme en langage algorithmique à une machine à mémoire effaçable.

```
Début
Initialiser a[1..n] ; v
v →b[n]                          {mémoriser la valeur v dans b[n]}
Pour i de n à 2
        b[i]*v → b[i-1]          {calculer chaque terme v^i et mémoriser dans b[i-1]}
Pour i de 1 à n
        b[i]*a[i] → a[i]         {calculer chaque terme a[i]*v^i et mémoriser dans a[i]}
Pour i de 1 à n
        calculer la somme des b[i]   {calculer la somme de tous les termes}
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?

Il faut (n – 1) multiplications pour la première boucle et n pour la deuxième. La troisième boucle nécessite n additions. Au total, le coût en opérations est de (2n – 1) multiplications et n additions.

Concernant la mémoire, il faut deux tableaux de dimension a[1..n] et b[1..n].

Comparaison de l'algorithme de calcul direct avec l'algorithme de calcul sur les puissances

L'amélioration apportée par le deuxième algorithme qui consiste à calculer la puissance v^i par la formule récurrente $v^i = v^{i-1}.v$, réduit le nombre de multiplications de $\frac{n(n+1)}{2}$ à 2n – 1. Autrement dit le temps de calcul passe de l'ordre n^2 à l'ordre n.

Par contre le coût en mémoire reste le même : deux tableaux à une dimension a[1..n] et b[1..n] représentant n places dans la mémoire.

Or souvent dans l'écriture d'un programme à une machine à mémoire effaçable l'économie en termes de mémoire et de nombre d'opérations élémentaires est prise en compte. Le deuxième tableau pour les deux algorithmes est réservé exclusivement à mémoriser les résultats intermédiaires.

On peut alors chercher à améliorer l'algorithme de calcul, en termes de nombre d'opérations et de mémoires, autrement dit un travail mathématique sur la formule de calcul s'impose.

II.1.3. Algorithme de Hörner

La formule de la fonction polynomiale est transformée de façon à se ramener au calcul de la valeur d'une succession de fonctions affines (fonction polynomiale de degré 1) imbriquées les unes dans les autres :

$$f(x) = ((\ldots(a_n x + a_{n-1})x + \ldots)x + a_o$$

Le calcul de la valeur de f(x) en la valeur v débute par le calcul de la valeur de la fonction affine la plus intérieure : $a_nx + a_{n-1}$, puis on multiplie le résultat par v, ajouter a_{n-1} au produit, ainsi de suite. Le dernier résultat obtenu ne doit pas être mémorisé, il est utilisé directement dans la suite du calcul.
Cet algorithme de calcul, appelé schéma de Hörner [1], peut être représenté comme suit :

Coefficients	a_n	a_{n-1}	…	a_0
Calculs	$a_n \rightarrow b_n$	$a_nv + a_{n-1} \rightarrow b_{n-1}$	…	$((\ldots(a_nv + a_{n-1})v + \ldots)v + a_o \rightarrow b_0$

On vérifie immédiatement que $b_i = a_nv^n + a_{n-1}v^{n-1} + \ldots + a_iv^i$.
Pour l'organisation des données, il faut mémoriser les coefficients dans un tableau à une dimension a[1..n].
Nous décrivons ce processus de calcul comme un programme en langage algorithmique à une machine à mémoire effaçable :

```
Début
Initialiser a[1..n] ; v
a[n] → b
Pour i de n  à 1
        a[i]*v + b → b                {calculer f(v)}
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?
Dans cet algorithme, il faut n multiplications et n additions.
Quant à la mémoire, il suffit d'un tableau à une dimension a[1..n] pour mémoriser les coefficients. La valeur de la variable après la boucle est la valeur de la fonction polynomiale pour v.
L'algorithme de Hörner est donc plus économique en termes de nombre d'opérations et en termes de mémoire que les deux algorithmes précédents (calcul direct et calcul direct sur les puissances) :

> Ce procédé de calcul de la valeur numérique d'un polynôme est très avantageux car il coûte seulement *n* multiplications et *n* additions alors que l'évaluation directe des puissances de x et des termes du polynôme coûte *2n – 1* multiplications et n additions. (Kuntzmann 1969, p. 27)

Signalons que l'algorithme de Hörner a été enseigné comme une méthode lors de l'introduction de l'informatique dans EMS au Viêt-nam (cf. chapitre A3). Il n'a jamais été officiellement enseigné dans EMS en France.

Est-elle la méthode la plus économique ?
En utilisant la notion de « chaîne polynomiale », Knuth (1969) montre qu'il existe un algorithme, dit par « adaptation des coefficients », qui peut évaluer un polynôme de degré n ($n \geq 3$) en utilisant au plus (m + 3) multiplications, m + 3 additions lorsque n est un nombre pair et exige (m + 2) multiplications et m + 2 additions lorsque n est un nombre pair où $m = \left\lceil \frac{n}{2} \right\rceil^2 - 1$ (cf. Knuth 1971, pp. 432-435).

[1] A noter que bien que cet algorithme de calcul soit souvent attribué à Horner W.G, mathématicien anglais qui l'a présentée dans *Philosophical Transaction*, Royal Society of London **109** (1819), c'est Newton qui l'a découverte cent ans plutôt et décrite dans *Analysis per Quantitatem Series* (London, 1711) (Knuth 1971, p. 423)
[2] $\lceil x \rceil$ est le plus petit entier parmi les entiers plus grands que x ; $\lfloor x \rfloor$ est le plus grand entier parmi les entiers plus petit que x.

II.2. Calcul des images par une fonction polynomiale de m nombres, m > 1

Rappelons le problème :

> Soit f une fonction polynomiale degré n. Calculer les images par cette fonction des nombres x_0, $x_1 \ldots x_{m-1} \ldots$ espacés d'un pas p et appartenant à l'intervalle [a, b] avec x_0 = a.

Nous allons présenter trois algorithmes, suivant que le calcul s'effectue à partir des valeurs de x (formule fonctionnelle ou formule récurrente) ou à partir des différences finies (algorithme de Babbage).

II.2.1. Algorithme utilisant la formule fonctionnelle $x_0 = a$ et $x_k = a + kp$

Le processus de calcul est le suivant :

- calculer les m valeurs de x, sachant que $x_0 = a$ et que $x_k = a + kp$
- algorithme de calcul de l'image d'un nombre v par une fonction polynomiale connue, v remplaçant successivement chacune des m valeurs trouvées : cet algorithme peut être l'un des trois algorithmes précédents

Description algorithmique du processus de calcul à une machine à mémoire effaçable :

```
Début
Pour i de 1 à m
  Calculer x = a + i × p
  [Calculer f(x)]          {l'un des trois algorithmes de calcul précédent}
Fin pour
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?

Pour le calcul de m valeurs de x sur [a, b] sachant que $x_0 = a$ et que le pas constant est p, cet algorithme nécessite une addition et une multiplication dans chaque boucle. Pour le calcul de la fonction en a + kp, d'après II.1, le coût de calcul est celui de l'algorithme choisi parmi les trois algorithmes étudiés (nombre de multiplications, nombre d'addition) : ($\frac{n(n+1)}{2}$, n), (2n – 1, n) et (n, n) dans chaque boucle.

Cet algorithme ne nécessite pas le stockage des valeurs dans une mémoire.

II.2.2. Algorithme utilisant la formule récurrente $x_0 = a$ et $x_{k+1} = x_k + p$

Le processus de calcul est le suivant :

- calculer m valeurs de x, à commencer par $x_0 = a$ suivant la formule $x_0 = a$ et $x_{k+1} = x_k + p$
- algorithme de calcul de l'image d'un nombre v par une fonction polynomiale connue, v prenant le m valeurs trouvées : cet algorithme peut être l'un des trois algorithmes précédents

Description algorithmique du processus de calcul à une machine à mémoire effaçable :

```
Début
a → x
Pour i de 1 à m
  x + p → x
  [Calculer f(x)]          {l'un des trois algorithmes de calcul précédent}
Fin pour
Fin
```

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?

Pour le calcul de m valeurs de x sur [a, b] sachant que $x_0 = a$ et que le pas constant est p, cet algorithme nécessite une addition dans chaque boucle. Le coût du calcul de l'image est,

comme dans l'algorithme précédent, celui de l'algorithme choisi parmi les trois algorithmes étudiés.
L'algorithme utilisant la formule récurrente de x nécessite le stockage de la valeur de x dans une mémoire. Cette valeur changeant au cours du calcul, cette mémoire est une mémoire variable.

II.2.3. Algorithme utilisant les différences finies

On peut adapter l'algorithme des différences finies de Babbage pour calculer successivement les valeurs exactes des images des m nombres (cf. II).

Quel est le coût du calcul en nombre d'opérations et en termes de mémoire ?
Le calcul des m images de m nombres $x_0 = a$, $x_1 \ldots$, x_{m-1} nécessitent $n(m - (n + 1))$ additions. En termes de mémoire, il faut définir n variables informatiques, c'est-à-dire n emplacements dans l'unité de mémoire.

Conclusion

Pour notre ingénierie didactique, nous avons choisi un problème de tabulation de la deuxième classe de problèmes qui, comme nous l'avons montré, joue un rôle central pour le problème de tabulation d'une fonction.

> Soit **f** une fonction polynomiale degré **n**. Calculer les images par cette fonction des nombres x_0, $x_1 \ldots x_{m-1} \ldots$ espacés d'un pas **p** et appartenant à l'intervalle **[a, b]** avec $x_0 = a$.[1]

Le problème mathématique est présent dans les deux institutions, France et Viêt-nam, mais l'usage des machines pour les calculs effectifs est fort différent :
- en France, on attend le plus souvent des calculs à la main mais aussi à l'aide du tableur de la calculatrice ou des logiciels tableurs ; cependant l'instrument prend alors en charge la répétition des calculs et son usage évite la formulation de la condition d'arrêt de cette répétition :

> Cet usage décharge l'élève de la répétition des calculs et masque l'intervalle sur lequel on fait le calcul, et qui, cependant, conditionne son début et sa fin. Cet usage ne rend-il pas difficile la formulation de la répétition des calculs par l'écriture d'une boucle ? Ne s'oppose-t-il pas à la nécessité de la formulation de l'initialisation et de la condition d'arrêt de cette action répétée ? (Nguyen et Bessot 2003, p. 16)

- au Viêt-nam, le tableau des valeurs est effectué officiellement à la main et seulement pour quelques valeurs particulières donnant lieu au calcul (à la main) des valeurs exactes pour les images.

Notre projet est d'utiliser une situation comportant un problème de tabulation non problématique (dont une technique de résolution est déjà bien connue) dans laquelle il est nécessaire de faire exécuter la tabulation par une machine à mémoire effaçable ou non, l'exécution à la main étant rendue trop coûteuse par le grand nombre de calculs identiques à répéter : une variable didactique du problème est alors la valeur du pas p relativement à la longueur de l'intervalle. Si le pas est petit par rapport à la longueur de l'intervalle [a, b], le nombre de calculs des images devient très grand.

La communication de la solution à une machine à mémoire effaçable ou non (deuxième composante de la situation fondamentale) a pour enjeu de déléguer à cette machine les calculs répétitifs et passe nécessairement par l'écriture d'un programme de calcul.

[1] Nous avons indiqué en gras les variables didactiques du problème.

Si la longueur d'un programme reste égale au nombre de calculs, cette écriture peut s'avérer très coûteuse, voire impossible. L'écriture économique d'un programme requiert l'usage des notions de variable et de boucle, elles-mêmes intrinsèquement liées à l'effaçabilité de la mémoire de machine.

La machine minimum choisie est une calculatrice, mais quelles calculatrices sont présentes dans EMS ?
Quels types de tâches mathématiques sont instrumentés par la calculatrice ? En particulier, la tabulation d'une fonction numérique l'est-elle ? Comment ?
Quelles caractéristiques de la calculatrice peuvent permettre de poser le problème de l'écriture d'un algorithme itératif ?
Le chapitre suivant va chercher à répondre à ces questions.

Chapitre C2
Quelle machine dans EMS ?
Vers la conception d'une calculatrice générique Alpro

Introduction

Comme annoncé et justifié dans l'introduction générale à notre travail, nous restreignons l'étude dans EMS aux calculatrices scientifiques, qu'elles soient programmables ou non programmables.
Quelle est l'architecture générale de cette machine calculatrice ? Comment la placer par rapport à celle de la machine de Von Neumann ?
Avec quels langages communique-t-on avec elle ?
Une première partie de ce chapitre cherche à répondre à ces questions.
Nous avons montré que les mémoires peuvent être des candidates pour donner du sens à la notion de variable informatique. Pour cette raison, nous allons dans une deuxième partie nous intéresser aux mémoires des calculatrices présentes dans EMS : quelle place occupe ces mémoires dans EMS ? Sont-elles un enjeu didactique pour EMS ?

I. Les touches mémoires des calculatrices de EMS

Par abus de langage et par commodité, nous appelons touches mémoires les touches des calculatrices qui renvoient à une place dans l'unité de mémoire de la machine calculatrice. Pour une analyse plus complète des différents types de touches mémoires des calculatrices présentes dans EMS (niveau collège et lycée) nous renvoyons le lecteur à l'annexe C2.
Ici, nous nous contentons de résumer dans trois tableaux les types de mémoires, associées à des touches, des calculatrices non programmables présentes dans EMS au Viêt-nam (tableau 1) et en France (tableau 2), ainsi que celles des calculatrices programmables présentes au Lycée en France (tableau 3).

Hiérarchie des modèles : du plus simple au plus complexe	Fx 220	Fx 500A	Fx 992	Fx 500MS	Fx 570MS
Mémoires A, B, C et nombre de ces mémoires			× 7	× 9	× 9
Mémoire du dernier résultat Ans			×	×	×
Mémoire M+, M-	×	×	×	×	×
Mémoire d'expression				×	×
Mémoire de formule				×	×
Parenthèses	×	×	×	×	×
Mémoire constante	×	×	×		
Touche X ↔ Y	×	×	×		
Touche X ↔ M	×	×			
Système S.V.P.A.M[1]			×	×	×

Tableau 1. Présence des mémoires selon le modèle de calculatrices non programmables (Viêt-nam)

[1] Le système S.V.P.A.M, acronyme de : "Super Visually Perfect Algebraic Methode", permet d'entrer une expression sur la calculatrice à l'identique de son écriture algébrique usuelle. Exemple : si l'on veut calculer sin 30°, on tape « sin » « 30° ». Sur celle qui ne possède ce système, on doit taper « 30° » « sin ».

Le tableau 1, montre la présence de calculatrices performantes comme Fx500MS, Fx570MS dans EMS. En effet, depuis 2001, le ministère de l'éducation et de la formation du Viêt-nam (MEF) encourage leur usage.

Hiérarchie des modèles : du plus simple au plus complexe[1]	Fx 92 Collège	TI 15	Fx Junio	TI 30X IIB	TI 40 Collège	HP 30S	Fx 92 + Collège
Mémoire A, B, C et nombre de ces mémoires				× 5	× 5	× 9	× 8
Mémoire Ans				×	×	×	×
Mémoire M+, M-	×	×	×			×	×
Mémoire d'expression				×	×	×	×
Mémoire de formule				×	×	×	×
Parenthèses	×	×	×	×	×	×	×
Mémoire constante	×		×	×	×	×	×
Touche X ↔ Y	×		×				
Touche X ↔ M	×		×				
Système S.V.P.A.M					×	×	×

Tableau 2. Présence des mémoires selon le modèle de calculatrices non programmables (France)

Modèle	Sharp EL - 9400	Fx 100 +	HP	TI 92
Mémoire A, B, C et nombre de ces mémoires	× 26	× 28	× 26	× 26
Mémoire Ans	×	×	×	×
Mémoire M+, M-				
Mémoire d'expression		×	×	×
Mémoire de formule	×	×	×	×
Parenthèses	×	×	×	×
Mémoire constante				
Touche X ↔ Y et X ↔ M				
Système S.V.P.A.M	×	×	×	×

Tableau 3. Présence des mémoires selon le modèle de calculatrices programmables (Lycée, France)

En France on rencontre le plus souvent les deux marques, Casio et Texas Instruments, alors qu'au Viêt-nam la marque Casio a un quasi monopole. En effet la marque Casio est fortement recommandée par l'institution et mentionnée presque exclusivement par les manuels vietnamiens.

Les trois tableaux montrent que les calculatrices les plus récentes présentes dans les deux pays possèdent des touches mémoires A, B, C, et ANS. Cependant dans ces calculatrices, le nombre de touches de mémoires A, B, C est plus important en France qu'au Viêt-nam : de 26 à 28 contre 7 à 9.

Quel est alors le lien entre ces mémoires « visibles » (rendues visibles par les touches associées) avec l'architecture générale d'une calculatrice ? Quels sont les langages utilisés ? Quelles sont les caractéristiques des touches mémoires A, B, C et Ans présentes dans toutes les calculatrices ?

II. Architecture et langages d'une calculatrice

Les calculatrices présentes dans EMS sont de deux types : non programmable ou programmable.

[1] Nous n'avons considéré que les deux marques Casio et Texas de calculatrices, car ce sont celles qui figurent les plus souvent dans les manuels. Les calculatrices non graphiques et non programmables sont utilisées plutôt en collège, les calculatrices graphiques et programmables, au Lycée.

Dans le cas d'une calculatrice non programmable, la suite des actions est la suivante : entrée au clavier d'une instruction de calcul lisible sur l'écran (ligne d'instructions ou ligne d'expression), demande d'exécution, lecture du résultat sur l'écran (ligne d'état). Durant le calcul s'affichent sur la ligne d'état, le mode de calcul (exact ou approché), le mode d'accès aux touches tapées (direct ou indirect), et un témoin d'activation de chaque touche mémoire.
Dans le cas d'une calculatrice programmable : entrée au clavier d'un programme (ensemble d'instructions) lisible sur l'écran de la calculatrice. La calculatrice l'enregistre dans la mémoire (programme en langage machine) puis l'exécute. Le résultat du programme est lisible sur l'écran. C'est le fonctionnement de la machine ordinateur de Von Neumann (chapitre B2). Comme l'unité de mémoire d'un ordinateur, les « mémoires » d'une calculatrice sont commandées par le processeur de la calculatrice et implémentées dans les registres de la mémoire centrale. Ces registres comportent une partie commandée par des touches-mémoires listées dans les tableaux 1, 2 et 3 précédents. On peut à tout moment avoir accès au contenu de ces mémoires. Nous les appelons « mémoires visibles » au sens où elles sont commandées par des touches de la calculatrice. La mémoire non visible correspond aux opérations ou instructions traduites en langage machine.

II.1. Architecture d'une calculatrice

L'architecture générale d'une calculatrice est schématisée dans la figure 1 ci-après :

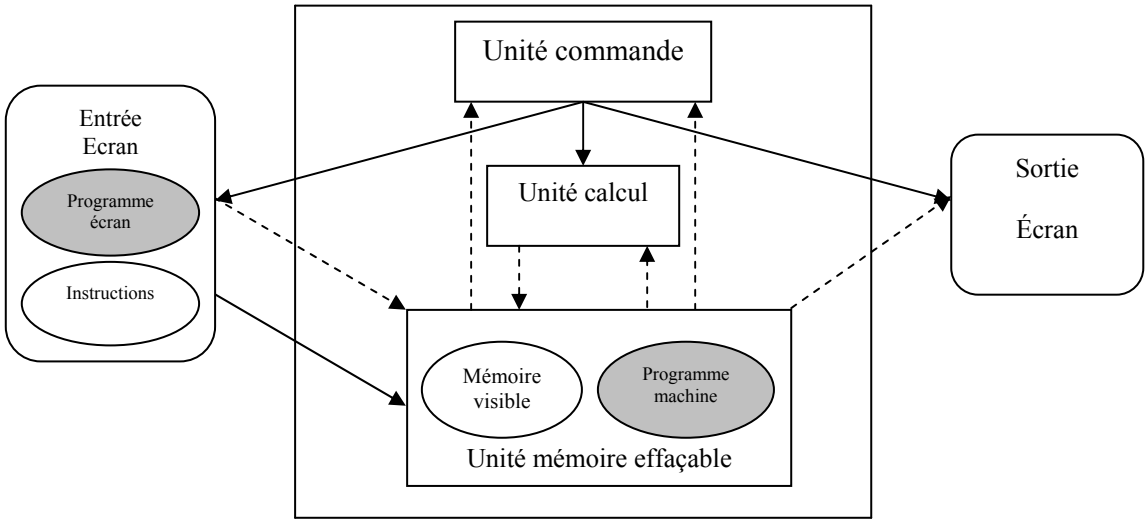


Figure 1. Architecture d'une calculatrice (programmable ou non programmable)

Légende : ⟶ : Instructions ; ⇢ : Transfert d'information

Dans ce schéma, nous avons grisé les parties spécifiques à la calculatrice programmable par rapport à une calculatrice non programmable.
Une calculatrice programmable est une machine de Von Neumann.

II.2. Langage de la calculatrice

Pour la communication entre l'homme et machine, les calculatrices donnent lieu à deux types de langages selon qu'elles sont non programmables ou programmables.

II.2.1. Deux types de langages

Nous considérons tout langage de calculatrice (ayant le système S.V.P.A.M) comme un langage évolué dans le sens où ce langage peut être compris directement par l'homme mais pas par la machine sans un langage intermédiaire.

Rappelons qu'un langage évolué peut être traduit par deux types de traducteur : compilateur et interpréteur (chapitre 3, partie B).
Le compilateur traduit un programme en langage évolué en un programme en langage machine. Le programme en langage machine est équivalent au programme source.
L'interpréteur traduit au fur et à mesure les instructions du programme en langage évolué. Les instructions traduites sont donc exécutées au fur et à mesure du processus de traduction.
Une calculatrice non programmable possède un interprétateur alors qu'une calculatrice programmable possède en même temps les deux types de traducteur.
Nous schématisons les deux processus correspondant à ces deux types de traducteurs dans la figure 2 ci-après.

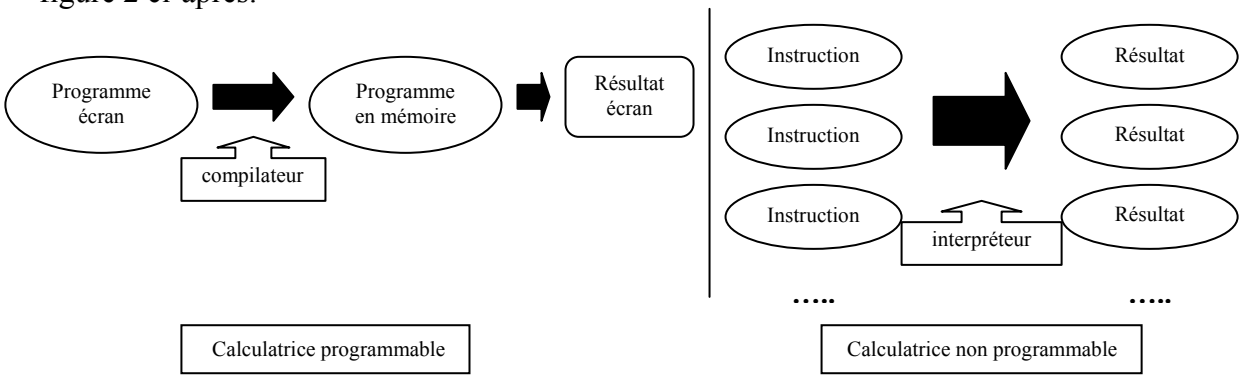


Figure 2. Deux types de processus de traduction du langage évolué des calculatrices

II.2.2. Rétroactions entre la calculatrice non programmable et l'utilisateur

Dans le cas d'une calculatrice non programmable, l'utilisateur écrit une instruction comme suite de touches de sa calculatrice, active l'exécution par la calculatrice et obtient immédiatement une information sur l'exécution de cette instruction : résultat ou message indiquant l'erreur (sémantique ou syntaxique) sur l'écriture de cette instruction. Cette situation induit un milieu (matériel) qui renvoie des informations au fur et à mesure des actions de l'utilisateur : ces informations valident ou invalident l'écriture de ses instructions. C'est donc un milieu pour la validation (au sens de Margolinas 1993) pour l'écriture des instructions de calcul (et non pour le calcul lui-même).

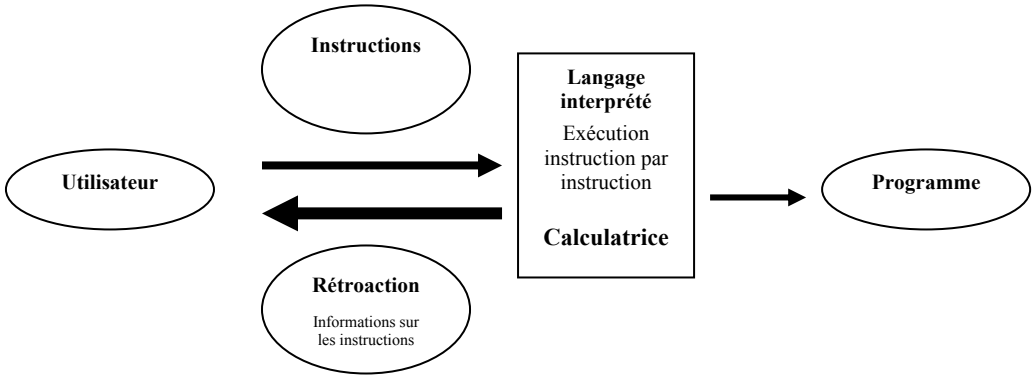


Figure 3. Exécution d'un programme sur une calculatrice avec langage interprété

Quels ajouts à une calculatrice non programmable pour qu'elle ait les fonctionnalités d'une machine de Von Neumann ? Les résultats des analyses précédentes (partie B) nous permettent d'identifier trois ajouts fondamentaux.

- Premier ajout à l'unité de commande : possibilité de commander et de combiner les trois structures fondamentales : séquentialité, branchement et itération contenus dans un programme qui lui est confié ;

- Deuxième ajout à l'unité de mémoire: possibilité d'enregistrer un programme en langage compris par l'unité de commande ;
- Troisième ajout à l'unité de commande et à l'unité de mémoire : capacité d'assurer l'interaction entre l'unité de commande améliorée et l'unité de mémoire améliorée.

Le premier ajout conduit à une machine ayant les fonctionnalités de la machine analytique de Babbage.
Les deux derniers ajouts marquent l'évolution la machine Analytique de Babbage vers la machine de Von Neumann.
Une calculatrice non programmable ne fonctionne ni sur le principe d'une machine analytique de Babbage, ni sur celui d'une machine de Von Neumann. Elle n'est pas une machine arithmétique non plus puisqu'elle possède des mémoires effaçables comme nous allons le voir maintenant.

III. Touches mémoires A, B, C et Ans

Notre analyse (chapitre B2) montre que le caractère d'effaçabilité d'une mémoire est nécessaire à l'émergence de la notion de variable. Rappelons que pour une telle mémoire le stockage d'une nouvelle valeur efface toute valeur déjà là.
Quelles touches mémoires visibles de la calculatrice peuvent devenir candidates à donner du sens à la notion de variable informatique ?

III.1. Effaçabilité et visibilité des touches mémoire A, B, C et Ans

III.1.1. Un premier type de mémoire : les touches A, B, C

Les opérations possibles sur ces touches mémoires sont :
+) Le stockage : il se fait soit par la touche « Sto » (TI) soit par la touche « → » (Casio)

Exemple : [5] [Sto] [A] ;[2] [+] [8] [×] [5] [Sto] [B] ;

+) Le rappel : le rappel du contenu d'une mémoire se fait par appui sur la touche mémoire suivi de celui sur la touche « Enter » chez TI (ou « = » chez Casio ou « Rcl » dans les deux marques).

Exemple : [A] [Enter] ; [A] [=] ; [Rcl] [A]

Dans les calculatrices de calcul formel et programmables, ces mémoires doivent être initialisées avant d'être utilisées. Dans les autres calculatrices, ces mémoires sont déjà initialisées à 0.
Quand une nouvelle valeur est stockée en mémoire, l'ancienne valeur est effacée.
Cette propriété d'effaçabilité nous amène à nommer ces mémoires « mémoires variables ».

Une variante en voie de disparition : les touches M+, M-

Les opérations possibles sur ces touches mémoires sont :

+) Le stockage : [5] [Min]

+) L'addition : [6] [M+] ; (respectivement, la soustraction : [2] [M-])

Cette opération met à jour le contenu mis en mémoire précédemment (ajouter 6 ou soustraire 2).

+) Le rappel : [MR] ;

Dans les calculatrices plus récentes, les touches [M+] et [M-] de cette mémoire ont disparu.

III.1.2. Un deuxième type de mémoire : la touche Ans

Une deuxième mémoire se distingue de la « mémoire variable » par le fait que le stockage dans cette mémoire suit automatiquement tout appui sur la touche « = » (ou « Enter »). *Cette mise en mémoire peut donc ne pas être le résultat d'un stockage volontaire et ce stockage*

peut même être ignoré de l'utilisateur de la calculatrice. En ce sens cette mémoire est moins visible que la précédente.
Cette mémoire enregistre aussi automatiquement la valeur stockée dans une mémoire variable après l'appui sur la touche « Sto » (ou « → »).

Le résultat enregistré automatiquement peut être rappelé en appuyant sur la touche « ANS » sous la condition express que les touches « = » (ou « Enter »), « Sto » (ou « → ») n'aient pas été réutilisées. En effet, dès que l'une des touches « = », « Enter », « Sto » ou « → » est pressée de nouveau, un nouveau contenu est enregistré et efface l'ancien contenu.
En ce sens, le stockage *automatique* de tout nouveau résultat dans Ans, qui a pour particularité d'effacer son contenu, la rend impropre pour supporter la notion de variable informatique. La mémoire Ans est en quelque sorte la mémoire du dernier résultat.

III.1.3. Visibilité dans un calcul des touches mémoires variables et Ans

Les touches mémoires sont « utilisables » et deviennent visibles dans un calcul.
Avec les touches mémoires variables, par exemple, la suite de touches sur Casio : « 3.145874 » « Sto » « A » « 4 » « × » « Alpha » « A » « + » « 2 » « = » donne sur la ligne d'expression : 4 × A + 2
La touche mémoire Ans peut apparaître dans un calcul sans obligation de la rappeler (contrairement aux mémoires variable).
Par exemple, soit à exécuter le calcul sur Casio l'expression 3.14512 × 9.49578 + 2. On peut le faire par deux successions de touches séparées par « = » : « 3.14512 » « × » « 9.14578 » « = » « + » « 2 » « = ». On obtient sur la ligne d'expression apparaissant sur l'écran : Ans + 2 sans que la touche mémoire « Ans » ait été pressée.

III.1.4. Un touche mémoire invisible : la parenthèse

Les touches parenthèses sont commandées par une mémoire « pile » invisible pour sauvegarder provisoirement des valeurs numériques et fonctions dans une expression entrée à l'écran.
Dans certaines calculatrices, comme l'HP35, il n'y a pas de touches parenthèses mais obligation de recourir à un langage basé sur la « notation polonaise inverse » (NPI). Par exemple, pour calculer l'expression $E = 7 + \frac{8}{4 \times 4 - 5}$, il faut taper successivement sur les touches : « 7 » « Enter » « 8 » « Enter » « 4 » « Enter » « 4 » « Enter » « * » « 5 » « Enter » « - » « + » « + ». Ce travail mathématique de réécriture, de l'expression numérique initiale en langage évolué NPI, permet au compilateur de la calculatrice la traduction en langage machine.
Pour calculer la même expression E, le blocage des deux touches parenthèses d'une calculatrice de type Casio ou TI oblige à l'usage des touches mémoires visibles par exemple comme suit : « 4 » « x » « 4 » « - » « 5 » « = » « Sto » « A » « 7 » « + » « 8 » « ÷ » « A ».
Nous reviendrons plus loin sur ce recours obligé.

III.3. Accès aux touches mémoires variables et Ans

Nous nous référons à la troisième des contraintes données par Trouche (1996) pour classer les « contraintes instrumentales ».

1. Les contraintes internes : elles recouvrent les contraintes de modalités d'existence et les contraintes liées à l'univers interne. Ces contraintes recouvrent aussi celles liées à la nature de l'écran : les pixels (ce qui est « interne » ne coïncident pas nécessairement avec ce qui « ne se voit pas ») ;
2. Les contraintes de commandes : c'est ce que Rabardel entend par contraintes de finalisation, liées aux opérations que l'outil autorise : y a-t-il une commande dédiée au calcul de limite par exemple ? Il s'agit d'une partie des contraintes liées à l'interface.

> 3. Les contraintes d'organisation : l'existence d'une commande n'est pas tout : il faut aussi considérer l'accès à cette commande, comparer les différents accès… Il s'agit de l'organisation de l'interface qui contribue d'une certaine façon à « *pré structurer l'action de l'utilisateur* » (mais ce n'est pas le seul élément de pré structuration : l'existence même des commandes est aussi une pré structuration). Les contraintes de commande regroupent aussi une partie des contraintes liées à l'interface. (Trouche 1996, p. 128)

Selon les calculatrices, l'accès aux touches mémoire variable et Ans peut être « directe » ou « indirecte », c'est-à-dire être la première ou la deuxième fonctionnalité d'une même touche. Examinons successivement la mémoire Ans et les mémoires variables.

Mémoire Ans :
+) Direct : Dans certaines calculatrices, essentiellement de la marque Casio (Fx 500Ms, Fx 570 Ms, Fx 6800) cette touche mémoire est la première fonctionnalité de la touche.
Exemple : la suite de touches sur Casio fx-570MS « 2 » « x » « 3 » « = » ; « Ans » « + » « 5 » donne le résultat 11
Dans d'autres calculatrices à la place de la touche « = », on a, soit « Exe », soit « Enter ».
+) Indirect : Dans d'autres calculatrices, essentiellement de la marque TI et Sharp (TI92, Sharp 9400) cette touche mémoire est la deuxième fonctionnalité de la touche.
Exemple : la suite de touches sur TI92 « 2 » « x » « 2 » « = » ; « 2nd » « + » « 5 donne le résultat 11.
Mémoire variable :
+) Direct : Dans d'autres modèles les plus performants, par exemple TI 92, on peut appuyer directement sur ces touches mémoires pour y avoir accès.
Exemple : supposons que la valeur 5 soit stockée dans la mémoire variable A. La suite de touches qui donne le double du contenu de cette touche mémoire est : « 2 » « x » « A ».
+) Indirect : Dans la plupart des calculatrices (Casio Fx, Sharp, TI89) il faut appuyer sur une touche spéciale « Alpha » pour avoir accès à ces touches mémoires (souvent désignées par des lettres alphabètes comme A, B, C)
Exemple : La suite précédentes de touches devient : « 2 » « x » « Alpha » « A ».
Dans les deux cas, on peut rappeler le contenu avec la touche « Rcl » qui est, elle-même, manipulée directement ou indirectement (avec la touche « Shift » ou « 2^{nd} »).

Mais quel rapport les élèves ont-ils à la calculatrice, en particulier, aux touches mémoires visibles de la calculatrice ? Jusqu'à quel point ces touches mémoires sont-elles instrumentalisées dans les tâches institutionnelles de calcul numérique ?

IV. Place et rôle de la calculatrice à mémoires dans EMS en France et au Viêt-nam

IV.1. Au Viêt-nam

Absence institutionnelle de la calculatrice

Dans son étude sur la vie de la calculatrice dans le programme des mathématiques actuel au Viêt-nam, Nguyễn T.N.H (2004) repère deux niches principales de cet instrument de calcul : « vérification des résultats d'un calcul » et « aide aux calculs ».

> Dans l'institution scolaire au Viêt-nam, les fonctions de la CDP [calculatrice de poche] sont les suivantes : vérification du résultat des opérations et aide aux calculs. La première fonction est plus marquée que la seconde qui, malgré son existence, semble être plutôt cachée. (Nguyễn T.N.H 2004, p. 55).

Cependant son analyse des exercices des manuels sur le collège et le début du lycée (classe 9) montre la quasi absence de type de tâches de calcul instrumentées par la calculatrice au profit de l'usage des tables :

> En classe 6 : […] la CDP est considérée comme un instrument pour trouver les résultats des opérations arithmétiques. Toutefois, il n'y a qu'une présentation de ce rôle et non l'entraînement à l'emploi de la CDP.
> En classe 7 : […] on ne demande pas aux élèves de se servir de la CDP pour les calculs.
> En classe 8 : […] l'accent est toujours mis sur la table des formules trigonométriques, et la question de la CDP n'est pas abordée.
> En classe 9 : […] Elle n'est présente que dans cette leçon [« Nombre réels – Racine carrée »], et son rôle n'est qu'accessoire auprès des tables. (Nguyễn T.N.H 2004, pp. 10-14)

L'examen des récents sujets de Brevet (2004) atteste de l'absence complète de calculs instrumentés, tous les calculs étant exacts et algébriques.
À partir de 2001 le programme cadre, écrit par le Ministère de l'Éducation et de la Formation (MEF), réserve 3 séances à la fin de chaque année des classes du lycée à la rubrique intitulée « Pratique du calcul avec la calculatrice de poche fx-500A » avec la recommandation :

> Il faut utiliser la calculatrice tout au long du programme, l'enseignement doit profiter des moments pertinents pour expliquer à l'élève l'emploi de la calculatrice (Programme 2000)

Cette recommandation n'est accompagnée d'aucune suggestion d'activité. Comment les auteurs des manuels l'interprètent-ils ? Le souci du manque de matériel sert de prétexte aux auteurs pour renvoyer à une autre discipline, la physique, l'usage de la calculatrice dans EMS :

> L'emploi de la CDP pour résoudre les opérations concernant les erreurs, les équations et les inéquations comportant des coefficients décimaux est très répandu dans le monde. Toutefois, comme les élèves vietnamiens n'ont pas toujours les moyens pour s'en procurer, ils doivent s'entraîner durant les séances de physique. (Trần V.H et Văn N.C 2000, rédacteurs en chef des manuels du Lycée)

Dans les manuels du lycée, aucun exercice ne concerne le calcul instrumenté par la calculatrice comme le confirme Nguyễn T. N. H :

> En classes 11 et 12, il n'y a pas non plus mention de la CDP. Ce point de vue a conduit les auteurs de manuel à « éviter » de se référer explicitement à la CDP dans les objets d'enseignement concrets. (Nguyễn T.N.H 2004, p. 14)

L'examen des sujets du Baccalauréat et du concours d'entrée à l'université de ces dernières années (2004, 2005) montre qu'aucune question ne porte sur le calcul instrumenté. Ceci ne peut que renforcer l'absence des tâches concernant la calculatrice dans EMS.
Comme nous avons analysé (cf. chapitre A3 et annexe A3-1), la calculatrice est aussi absente des tentatives d'introduction de l'informatique dans l'enseignement secondaire (en tant que discipline « informatique » ou au sein des mathématiques).
Ces analyses nous permettent de conclure à l'absence institutionnelle (des programmes et des manuels) de la calculatrice dans EMS du Viêt-nam.

Mais présence dans les pratiques des élèves et des enseignants sans enjeu didactique

Cependant ce n'est pas pour autant que la calculatrice est absente des pratiques des enseignants et des élèves.
Une pré-expérimentation que nous avons menée dans des classes 10 et 11 de deux lycées de Hanoi[1] montre que la plupart des élèves (130/152 soit 85,5%) possèdent et utilisent des calculatrices de la marque Casio (fx 500A, fx 500 MS et fx 570 MS). Une enquête menée par Lê Thái Bảo (2004) auprès d'enseignants d'Ho Chi Minh ville le confirme :

[1] En mai 2003, dans les Lycées Chu Văn An et Kim Liên

- La plupart des enseignants de collèges autorisent leurs élèves à utiliser la calculatrice pour calculer les rapports trigonométriques et extraire les racines carrées et les racines cubiques. Par contre, les tables numériques (trigonométrique, racines carrées et cubiques) sont rarement utilisées.
- Au début de la classe 10, beaucoup d'élèves possèdent et savent utiliser la calculatrice Casio fx 500A […] (Lê Thái Bảo 2004, p. 34)

La calculatrice existe bien comme instrument de calcul chez l'élève et pour les enseignants dans un cours des mathématiques, au moins dans les deux grandes villes du pays.
Cette présence est prise en compte par le MEF. Dans un arrêté récent (2005), le MEF autorise l'usage des calculatrices au baccalauréat et au concours d'entrée à l'université et précise les types de calculatrices permis :

Le candidat est autorisé à amener dans la salle d'examen :
- Stylo, règle, crayon noir, compas, équerre, outils pour la construction géométrique.
- Calculatrice ne possédant pas la fonctionnalité de traitement de texte et la carte mémoire. Plus précisément, sont autorisées les calculatrices ne pouvant réaliser que les quatre opérations arithmétiques, la racine carrée, et la puissance ; sont autorisées les calculatrices de la marque Casio fx95, fx200, fx 500A, fx 500MS, fx 570MS et les calculatrices équivalentes (avec les touches trigonométriques, logarithmes, exponentielles) (MEF, décret n° 3343/KT&KD 29/04/2005)

Ces constats conduisent à questionner les pratiques : comment la calculatrice est-elle utilisée ? En particulier, les touches mémoires sont-elles prises en compte dans l'usage de cet instrument de calcul ? Lesquelles ? Comment sont-elles instrumentalisées ?

IV.2. Place et rôle de la calculatrice à mémoires dans EMS en France

Présence institutionnelle forte de la calculatrice

Depuis la contre réforme des années 80, on peut attester de la place importante accordée à la calculatrice par le programme afin de promouvoir le caractère expérimental des mathématiques. La présence du nouvel instrument de calcul, la calculatrice, a fait disparaître les anciens instruments de calcul : tables de logarithme, règle à calcul :

L'utilisation systématique des calculatrices, qui dispense naturellement d'avoir recours aux instruments antérieurs (table de logarithmes, règle à calcul,…) constitue un nouveauté du programme de mathématiques. […] Dès le début de l'année, il sera bon de vérifier que chacun sait utiliser son propre instrument, et ce sera une occasion de préciser l'usage des parenthèses et de réviser les propriétés de R (programme 1982)

Dans les années 90, les calculatrices, programmables et graphiques, occupent une place importante dans EMS avec deux fonctions, produire des calculs effectifs et vérifier des résultats, qui figurent clairement dans les recommandations du programme :

Les problèmes et les méthodes numériques doivent eux aussi tenir une large place. L'emploi systématique des calculatrices scientifiques renforce les possibilités d'étude de ces questions, aussi bien pour effectuer des calculs que pour vérifier des résultats ou alimenter le travail de recherche (programme 1989)

Voici quelques recommandations du programme 2000 à propos de la place de calculatrice en analyse :

- Utiliser de façon raisonnée et efficace la calculatrice pour les calculs et pour les graphiques ; calculatrice et grand nombre ; organiser un calcul à la main ou à la machine ; (classe Seconde) ;
- Calcul des termes d'une suite sur calculatrice ou tableur (classe Première) ;
- On pourra approcher la solution de l'équation f(x) = k par dichotomie ou balayage avec la calculatrice ou au tableur (classe de Terminale) (programme 2000)

Ainsi l'incitation à recourir à la calculatrice pour produire des résultats effectifs pour deux types de calcul, non itératifs et itératifs, est explicitée par le programme. A noter la présence du tableur (de calculatrice ou d'ordinateur) dans les activités de calcul numérique elle aussi recommandée par le programme.

Avec la présence dominante des calculatrices programmables au lycée, une nouvelle niche apparaît pour la calculatrice, celle de « *mettre en œuvre les notions de boucle et de test* ».
Ainsi les programmes des années 70, 80, 90 et 2000 en France attribuent un certain nombre de fonctions à la calculatrice dans EMS en France : vérifier le résultat d'un calcul, réaliser des calculs effectifs, alimenter le travail de recherche et quant à l'aspect d'algorithmique et de programmation : favoriser la mise en œuvre des notions de base : boucle et test. Ceci concerne particulièrement les calculs itératifs.

Place faible accordée à l'usage de mémoires A, B, C et Ans dans le calcul

Que se passe-t-il dans les manuels ? Nous limitons notre analyse aux manuels des programmes des années 2000. Notre analyse des manuels (cf. chapitre A2) a montré que le travail de l'élève en ce qui concerne la programmation des calculs itératifs sur la calculatrice se réduit essentiellement à l'exécution de programmes tout faits sur la calculatrice.
Cependant, certains manuels, comme Déclic 1er S, Belin 1er S, recommandent l'usage de la touche mémoire Ans dans le calcul des termes d'une suite définie par récurrence.
En ce qui concerne les calculs non itératifs - calcul exact, calcul approché - une place importante est occupée par des exercices qui font appel à la calculatrice.
Dans ces exercices, quelle place occupe la touche mémoire variable comme A, B, C et Ans ? Quel usage en est attendu ?
L'examen de trois collections de manuels du lycée - Bréal, Belin et Déclic du programme 2000 (cf. annexe C2) – fait découvrir la forte présence du type de tâche C, « calculer pour un nombre donné la valeur numérique d'une expression à l'aide de la calculatrice » essentiellement en Seconde. Nous n'examinerons en détail que ce niveau.
Trois types de tâches proches de C sont en relation explicite avec les touches mémoires de la calculatrice.
M1 (mémoire variable) : « Calculer pour plusieurs nombres la valeur numérique d'une même expression en utilisant les mémoires A, B, C » Ce type de tâche relève d'un problème de tabulation d'une fonction, l'expression étant considérée comme la formule d'une fonction.
M2 (mémoire Ans) : « Afficher les chiffres cachés dans la mémoire d'une calculatrice en utilisant la mémoire Ans »
La différence de statut entre les touches mémoires A, B, C - mémoire variable - et celui de touche mémoire Ans - mémoire du dernier résultat- n'est mentionnée par aucun manuel.
Pour cette dernière, les manuels ne font pas la différence entre ce qui est à taper (la touche mémoire Ans) et ce qui apparaît sur l'écran (le symbole « Ans » apparaît sur l'écran).
Le travail mathématique nécessaire à la transformation de l'expression algébrique donnée dans l'énoncé pour l'entrée « écran » de la calculatrice est rendu présent par le type de tâche M3 suivant ; ce travail doit bien sûr tenir compte du langage de la machine.
M3 (mémoire parenthèse) : « écrire une expression mathématique en une suite de touches de la calculatrice en utilisant les touches parenthèses »
La touche mémoire « parenthèse » doit dans M3 outiller le travail de transformation.
Ce type de tâche donne lieu à d'autres sous-types de tâche comme « comparer l'écriture conventionnelle des calculs en l'écriture à l'écran d'une calculatrice » (Déclic).

Nous présentons, dans le tableau 4, le nombre d'exercices concernant les types de tâches M1 (mémoires variables), M2 (mémoire Ans) et M3 (mémoire parenthèse) selon les manuels de seconde étudiés.

Manuels	M1	M2	M3	Total
Belin	2	4	0	6
Bréal	0	0	0	0
Déclic	4	0	5	9
Total	6	4	5	15

Tableau 4. Nombre d'exercices de type Mi selon les manuels de seconde

Alors que le nombre d'exercices de type C « calculer la valeur numérique d'une expression en utilisant la calculatrice » est très important, on constate le faible nombre d'exercices de ce type qui demande explicitement l'usage des touches mémoires. Un indicateur du caractère facultatif de ces exercices est pour nous le fait que le manuel Bréal ne propose aucun exercice se rattachant à ces types de tâche.

En conclusion sur le statut de la mémoire d'une calculatrice dans les deux institutions

Bien que notre étude présente un paysage contrasté en ce qui concerne la place et le rôle de la calculatrice dans les deux institutions (lycée en France et au Viêt-nam), elle met en évidence une similarité des usages des mémoires dans les pratiques des calculs à l'aide de la calculatrice.
Ce constat nous conduit à formuler l'hypothèse de recherche suivante.

Hypothèse de recherche sur le fonctionnement de la calculatrice dans EMS

Dans EMS des deux pays, France et Viêt-nam, la calculatrice fonctionne comme une machine arithmétique dans les tâches de calcul, c'est-à-dire comme une machine sans mémoire effaçable.

Pour valider (ou invalider) cette hypothèse, il y a nécessité dans la conception de l'ingénierie didactique de construire une situation qui permette au chercheur d'accéder au rapport (le plus souvent privé) de l'élève à la calculatrice et de favoriser la transformation de ce rapport en instrumentalisant les touches A, B, C comme mémoires variables contre Ans, mémoire du dernier résultat.

V. Conception de la calculatrice Alpro

Les calculatrices à mémoire sont bien présentes dans EMS. Mais comment sont-elles utilisées ? En particulier, les touches mémoires sont-elles prises en compte dans l'usage de cet instrument de calcul ? Lesquelles ? Comment sont-elles instrumentalisées ? Comment peut-on avoir accès aux pratiques avec la calculatrice qui restent le plus souvent privées ? Comment peut-on rendre publiques ces pratiques ?
Pour avancer dans l'étude de ces questions et mettre à l'épreuve notre hypothèse de recherche sur le fonctionnement de la calculatrice dans EMS, nous avons décidé de définir un cahier des charges pour un projet, nommé Alpro (abréviation d'Algorithme et Programmation), que nous allons maintenant présenter.

V.1. Principaux objectifs du projet Alpro

Notre cahier des charges désigne trois objectifs principaux au projet Alpro.

1. Alpro est un émulateur de calculatrice générique

L'interface d'Alpro ne copie aucun modèle physique connu de calculatrice et possède potentiellement les mêmes fonctionnalités que les modèles scientifiques non graphiques et non programmables actuels.

2. Alpro génère automatiquement un historique des touches actionnées

Le logiciel de l'émulateur a la capacité d'enregistrer des séances dans un fichier externe destiné à subir des traitements statistiques sur son contenu. Le processus enregistre entre

autres les touches actionnées (touches d'opérations, touches de fonctions, touches mémoires, etc.), les calculs intermédiaires et les messages d'erreurs.

3. Alpro possède une boîte à outils à la disposition de l'enseignant

L'interface d'Alpro a la particularité d'être modulable par l'enseignant au sens où l'enseignant peut en modifier l'aspect fonctionnel ou visuel. La boîte à outils contrôle deux aspects fondamentaux de cette modularité :

- *Aspect graphique* : définition de l'ensemble des touches qui sont présentes et/ou disponibles sur l'interface (touches fonctions mathématiques, touches opérations de bases, etc.), et leurs comportements (activées / désactivées).
- *Aspect fonctionnel* : contrôle du fonctionnement global de la calculatrice, sa précision de calcul et de stockage, ainsi que des paramètres en relation directe avec la génération du ficher historique.

V.2. La première version d'Alpro réalisée

À partir de ce cahier des charges, une première version du logiciel a été développée en coopération avec Léhatem, étudiant en Master informatique de l'Université Joseph Fourier (Grenoble 1), lors de son stage de recherche effectué au sein de l'équipe DDM (2004).
D'après lui, les émulateurs existant sur ordinateur à l'heure actuelle peuvent être classés en deux catégories relativement au noyau de calculs :
- Émulateurs avec noyau de l'ordinateur qui utilise les primitives disponibles du système de l'ordinateur sur lequel il est installé.
- Émulateurs avec noyau chargé qui recourt à un noyau externe issu d'une calculatrice physique.

Nous avons décidé de :

- développer un émulateur avec noyau de l'ordinateur : il peut profiter des fonctionnalités du noyau de l'ordinateur sur lequel est installé l'émulateur.
- définir des fonctionnalités supplémentaires en adéquation avec le cahier des charges.
- développer cet émulateur en langage Java script car Java est conçu pour être un langage multi plate-forme :

> Ce logiciel pourra être installé sur n'importe quel ordinateur (PC et Mac) à condition qu'il soit doté d'une *Machine Virtuelle Java* (JVM). Les techniques XML et XPATH pour la formation des fichiers historiques ont aussi été utilisées. (Léhatem 2004, p. 32).

V.2.1. L'émulateur d'Alpro (objectif 1)

Examinons la version actuelle du point de vue des deux aspects graphique et fonctionnel.

Aspect graphique
L'émulateur présente une interface graphique avec laquelle l'utilisateur, enseignant ou élève, peut interagir. Il dispose d'un ensemble de touches regroupées par pavés et d'un écran pour visualiser les expressions construites et le résultat correspondant.

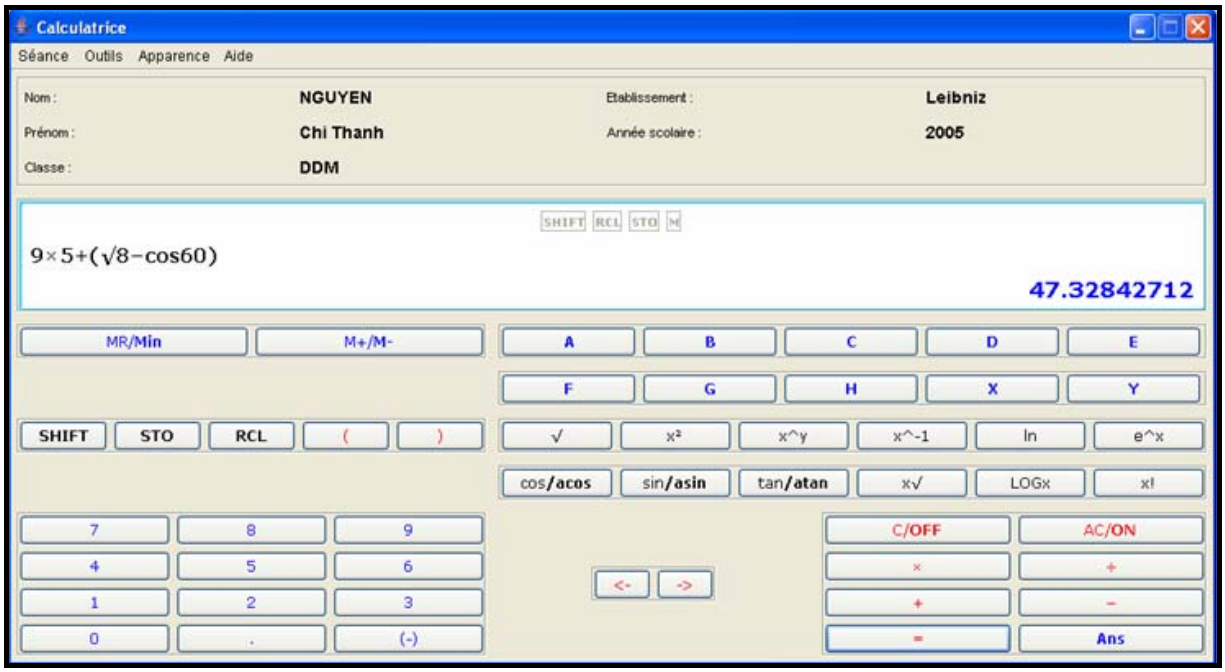


Figure 4. Aperçu global de l'émulateur (version non bloquée)

Les touches de l'émulateur se répartissent en quatre pavés :

- Touches de fonctionnement : C/OFF, AC/ON, SHIFT ;
- Touches de formation de nombres : les chiffres 0..9, le signe « - », la virgule « . »;
- Touches de fonctions : √, x^2, x^y, x^-1, ln, e^x, cos/acos, sin/asin, tan/atant, x√, LOGx, x! ;
- Touches de mémoire : les parenthèses et trois type de mémoire à savoir, « mémoire du dernier résultat » Ans, « mémoires variables » A, B, C et mémoire M, ainsi que les touches qui y sont liées Sto et Rcl.

L'écran de l'émulateur comporte trois lignes : ligne d'état, ligne d'expression et ligne de résultats.

Figure 5. Écran de l'émulateur actuellement développé

Aspect fonctionnel

L'émulateur respecte toutes les normes auxquelles les calculatrices dites scientifiques se soumettent comme la précision des calculs, les longueurs maximales des nombres à l'affichage ou dans le stockage, les règles d'arrondis des chiffres en virgule flottante, les priorités entre les différentes fonctions et opérations, etc.

Les nombres sont affichés à 10 chiffres sur l'écran et sont enregistrés à 14 chiffres dans les mémoires. L'arrondi des nombres se fait donc sur le 15^{éme} chiffre lors de stockage, et sur le 11^{éme} chiffre lors de l'affichage.

La règle d'arrondi des chiffres en virgule flottante est d'incrémenter le chiffre dans la position « i », si le chiffre qui se trouve à la position « i + 1 » est supérieur ou égale à 5.

Le niveau d'imbrication des parenthèses est fixé à 18 dans une expression mathématique à évaluer.

Les types de mémoire choisis pour cet émulateur sont : Ans, M+, A, B, C (mémoires visibles) et parenthèses (mémoire invisible). Nous avons laissé de côté les touches mémoires comme

mémoire constante, mémoire d'expression, car secondaires pour notre étude.
Une différence avec les calculatrices ordinaires tient dans notre décision d'obliger l'utilisateur à initialiser les mémoires A, B, C avant tout usage ce qui rend observable la prise en compte de l'existence d'une mémoire variable dans un programme de calcul. En cela notre calculatrice ressemble aux calculatrices de calcul formel ou programmables).
En ce qui concerne les contraintes d'organisation des touches mémoires, nous avons décidé de donner un accès direct aux touches mémoires A, B, C, Ans et aux touches de fonctionnement qui les concernent : Sto, Shift, Rcl., ceci afin de simplifier l'écriture des programmes de calcul.

V.2.2. L'historique des touches actionnées d'Alpro (objectif 2)

Avant toute utilisation de l'émulateur, l'utilisateur doit s'identifier pour accéder à la calculatrice ; cette identification déclenche l'enregistrement.
Le nom du fichier d'enregistrement de ses actions est du type « nom_prénom_numéro.xml ».
L'historique mémorise dans un *ordre chronologique* les touches actionnées, le contenu des mémoires, l'état de chacune des trois lignes de l'écran, dont l'expression mathématique écrite, le résultat du calcul ou l'éventuel message d'erreurs.

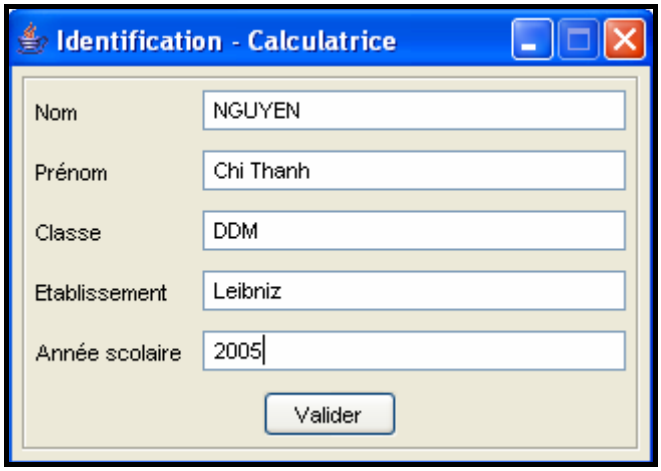


Figure 6. Fenêtre d'identification de l'utilisateur

On peut lire ce fichier avec un navigateur d'Internet comme IE. Les critères d'enregistrement sont le temps et les calculs intermédiaires.

> Le temps montre la progression et l'avancement de l'élève dans le travail, et puisque l'historique est basé sur l'enregistrement successif des touches actionnées, l'élève peut marquer un temps d'arrêt entre deux touches appuyées, ce temps d'arrêt est fixé à 15 seconds […] Si la durée entre deux touches actionnées est supérieure au seuil établi, on marque un nouvel enregistrement dans l'historique, sinon on actualise le dernier enregistrement. (Léhatem 2004, p. 12)

L'organisation des fichiers historiques est illustrée dans le schéma suivant (Léhatem 2004, p. 19) :

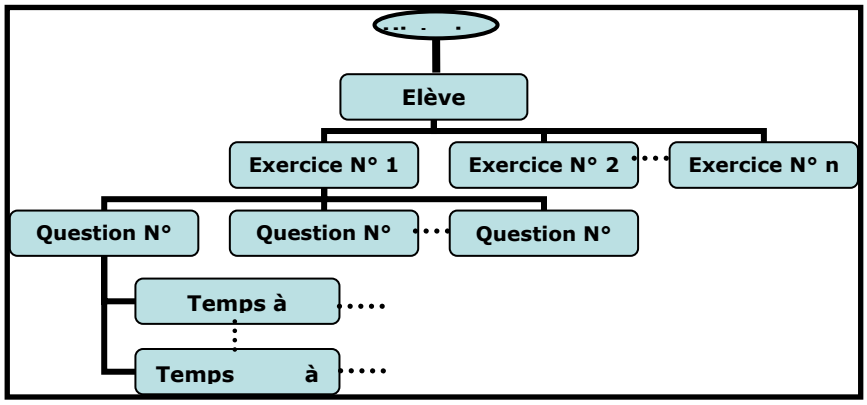


Figure 7. Organisation du fichier historique en arborescence XML

Nous donnons ci-après un extrait d'un fichier historique enregistré correspondant à la suite de touches tapée pour l'exécution du calcul 5 ÷ 9 :

```
- <Historique Date="11 juin 2004" Heure="17:26:53 CEST">
  - <Elève Nom="NGUYEN" prénom="Chi Thanh" Classe="2e D" Etablissement="GMD">
    - <Exercice Numéro="1" Date="11 juin 2004" Heure="17:27:02 CEST">
      - <Question Numéro="1" Date="11 juin 2004" Heure="17:27:07 CEST">
        - <Temps Date="11 juin 2004" Heure="17:27:32 CEST">
            <Etat_Calculatrice Etat="allumée" />
          - <Touches_Actionnées>
              <Touche Type_Touche="Touche Spéciale">AC/ON</Touche>
              <Touche Type_Touche="Touche opérande">5</Touche>
              <Touche Type_Touche="Touche Opération">÷</Touche>
              <Touche Type_Touche="Touche opérande">9</Touche>
            - <Touche Type_Touche="Touche Opération">
                =
              - <Calcul_Intermédiaire>
                  <Ligne_Expression>5÷9</Ligne_Expression>
                  <Ligne_Résultat>0.555555556</Ligne_Résultat>
                  <Ligne_Indicateurs />
                - <Etat_Mémoires>
                  - <Type_Mémoire Type="Mémoire Indépendante">
                      <Mémoire Nom="Ans">0.5555555555556</Mémoire>
                    </Type_Mémoire>
                  - <Type_Mémoire Type="Mémoire Variable">
                      <Mémoire Nom="A" />
                      <Mémoire Nom="B" />
                      <Mémoire Nom="C" />
                    </Type_Mémoire>
```

Figure 8. Un fichier historique

V.2.3. L'état de la boîte à outil d'Alpro (objectif 3)

L'émulateur se présente actuellement sous deux modes séparés : un mode non bloqué et un mode bloqué. Dans le premier mode, toutes les touches sont disponibles alors que dans le second, certaines touches apparaissant en grisé sur l'interface sont indisponibles *:*
- en ce qui concerne les touches mémoires : certaines touches mémoires variables : D, E, F, G, H, X, Y ; toutes les touches mémoires M+ ; toutes les touches mémoires parenthèses ;
- toutes les touches fonctions : x^2, √, x^y, x^-1, ln, e^x, cos/acos, sin/asin, tan/atant, x√, LOGx, x !.

Nous justifierons certains de ces choix dans la suite de notre travail.

V.3. Les développements prévus

L'émulateur au stade actuel de son développement ne remplit que partiellement chacun des trois objectifs du projet Alpro.
Voici les développements ultérieurs prévus :

- relativement à l'objectif 1 : un pavé statistique et un pavé « listes ».
- relativement à l'objectif 2 : des outils de traitement des fichiers de l'historique.

- relativement à l'objectif 3 : une véritable modularité sous la forme d'un panneau de configuration à la disposition de l'enseignant.

Résumé

Notre analyse de l'architecture des calculatrices montre qu'une calculatrice non programmable ne fonctionne ni sur le principe d'une machine analytique de Babbage, ni sur celui d'une machine de Von Neumann. Elle n'est pas non plus une machine arithmétique puisqu'elle possède des mémoires effaçables.
Les touches mémoires A, B, C sont associées à des mémoires visibles et effaçables, candidates à être le support de variables informatiques. La touche mémoire Ans est aussi associée à une mémoire visible et effaçable, mais son effaçabilité est très contrainte puisque tout nouveau résultat d'un calcul remplace automatiquement le contenu de Ans ce qui la rend impropre pour supporter la notion de variable informatique. La mémoire Ans est pour nous la mémoire du dernier résultat.
L'analyse des deux institutions (lycée en France et au Viêt-nam) prouve la présence de la calculatrice dans les pratiques des élèves et des enseignants, présence forte et officielle en France alors qu'elle reste officieuse au Viêt-nam. Dans les tâches de calcul à l'aide de la calculatrice, l'usage des mémoires reste quasi inexistant dans les attentes institutionnelles.

Ce constat nous a conduit à formuler une hypothèse de recherche.

Hypothèse de recherche sur le fonctionnement de la calculatrice dans EMS
Dans EMS des deux pays, France et Viêt-nam, la calculatrice fonctionne comme une machine arithmétique dans les tâches de calcul, c'est-à-dire comme une machine sans mémoire effaçable.

Pour mettre à l'épreuve cette hypothèse et avoir accès au rapport instrumental des élèves à la calculatrice nous avons conçu une première version d'une calculatrice générique, nommée Alpro, transportable et installable sur PC, Mac et Internet, et offrant les possibilités de récupérer des fichiers d'enregistrements des touches actionnées (historique). Cette calculatrice ressemble aux calculatrices non programmables à mémoires présentes dans EMS en France et au Viêt-nam.

Conclusion de la partie C

Introduire des objets d'algorithmique et de programmation dans le système d'enseignement actuel passe par des choix macro-didartiques et micro-didactiques en relation avec les hypothèses de la recherche qui seront mises à l'épreuve d'une réalisation didactique.

Dans cette partie, nous avons cherché à établir des choix macro-didactiques c'est-à-dire des variables de commande sur lesquelles nous prenons la décision d'agir et « *qui concernent l'organisation globale de l'ingénierie* » (Artigue 1988, p. 291)

Le premier choix est celui d'une situation fondamentale de l'algorithmique et de la programmation : en jouant sur les valeurs des variables de cette situation, on peut provoquer une co-genèse expérimentale des notions de variable informatique et de boucle à travers l'écriture de messages successifs à des machines dotées de caractéristiques différentes.

La situation fondamentale choisie articule le *problème mathématique de la tabulation d'une fonction numérique*, dont nous avons montré dans la partie B le rôle dans l'émergence et l'évolution des machines et des langages, et le *problème informatique de l'écriture d'un message à une machine à mémoire* pour une exécution effective d'une solution du problème mathématique.

Plus précisément, nous avons restreint le problème mathématique à la tabulation d'une fonction polynomiale :

> Soit **f** une fonction polynomiale degré **n**. Calculer les images par cette fonction de m nombres x_0, $x_1 \ldots x_{k-1}$ … espacées d'un pas **p** et appartenant à l'intervalle **[a, b]** avec $x_0 = a$.[1]

Ce problème de tabulation est routinier dans le cas où le degré de la fonction est inférieur ou égal à 2 et pour m petit. Une variable didactique cruciale de ce problème est la taille du pas relativement à la longueur de l'intervalle [a ; b], puisque cette taille détermine le nombre m : dans le cas où p devient très petit par rapport à (b-a), le calcul à la main de la tabulation devient très coûteux voire impossible dans un temps limité. La question de l'exécution de l'algorithme itératif, solution de ce problème, par une machine se pose alors.

Par quelle machine faire exécuter ce calcul ?
Nous avons décidé dès le départ, de choisir la machine minimale qu'est la calculatrice non programmable, présente dans les deux pays et à laquelle n'est attaché aucun langage de programmation.

L'analyse du chapitre 2 de cette partie a montré qu'une telle calculatrice ne fonctionne ni sur le principe d'une machine analytique de Babbage, ni sur celui d'une machine de Von Neumann et qu'elle n'est pas non plus une machine arithmétique. En effet, toutes les calculatrices actuelles présentes dans les deux institutions possèdent des mémoires effaçables (contrairement à une machine arithmétique). Parmi ces mémoires effaçables, nous avons distingué les touches mémoires A, B, C de la touche mémoire Ans, les premières étant les seules candidates à être les supports de variables informatiques.

[1] Nous avons indiqué en gras les variables didactiques du problème.

Mais la présence de touches mémoires n'assure pas leur instrumentalisation dans les pratiques des élèves. En effet, les techniques institutionnelles de calcul instrumentées par la calculatrice, en particulier celle de tabulation d'une fonction, ne requièrent pas l'usage de touches mémoires A, B, C.
Ce constat nous conduit à faire l'hypothèse qu'en fait la calculatrice fonctionne pour la plupart des élèves comme une machine arithmétique.

Hypothèse 5 (chapitre C2)
Dans EMS, la calculatrice fonctionne comme une machine arithmétique, c'est-à-dire une machine sans mémoire effaçable.

Si cette hypothèse est vérifiée, un enjeu de l'ingénierie didactique est de faire entrer les élèves dans une genèse expérimentale qui les conduise de la machine arithmétique qu'est la calculatrice non programmable à la machine de Von Neumann.

Nous allons dans la partie D finale présenter :

Les trois situations construites à partir de la situation fondamentale ;
Les conditions d'expérimentation du processus réalisé par l'articulation de ces trois situations ;
L'analyse a priori de chacune de ces situations et l'analyse a posteriori de leur réalisation ;
Des études de cas pour suivre qualitativement les évolutions de binômes d'élèves.

Partie D

Conception, réalisation et analyse

de l'ingénierie didactique d'introduction à l'algorithmique et la programmation dans EMS

Introduction

Les choix macro-didactiques faits pour l'ingénierie didactique, que nous allons présenter et analyser, sont les suivants :

Une situation fondamentale de l'algorithmique et de la programmation

Rappelons que cette situation articule un problème mathématique de tabulation : « Soit **f** une fonction polynomiale de degré **n**. Calculer les images par cette fonction de m nombres x_0, $x_1 \ldots x_k \ldots$ espacées d'un pas **p** et appartenant à l'intervalle **[a, b]** avec $x_0 = a$ » et le problème informatique de l'écriture d'un message à une machine à mémoire effaçable pour une exécution effective d'une solution du problème mathématique.

Une genèse expérimentale

La situation fondamentale a permis de concevoir l'ingénierie didactique comme une genèse expérimentale de la machine de Von Neumann et de la programmation à travers l'écriture des messages successifs (programmes) à des machines dotées de caractéristiques différentes. Nous schématisons dans la figure 1 ci-après les différentes machines de l'ingénierie didactique.

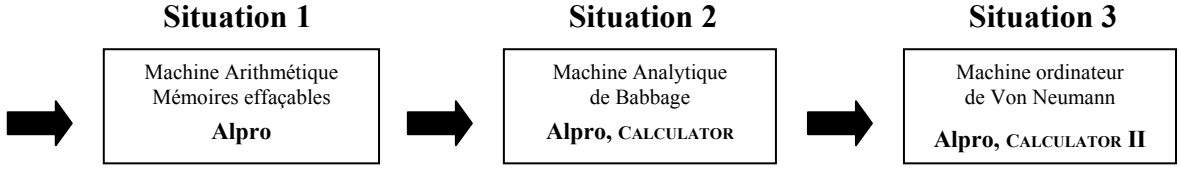


Figure 1. Les différentes machines dans les trois situations de l'ingénierie didactique

Un artefact commun aux trois machines est la calculatrice Alpro.

Le problème mathématique choisi, celui de la tabulation d'une fonction numérique, cherche à déclencher la fabrication de répétitions au travers de l'itération ou de la récursivité. Mais le nombre limité de mémoires d'Alpro privilégie l'algorithme itératif au détriment de l'algorithme récursif.

L'enjeu de la situation 1 est la familiarisation avec Alpro et ses différentes mémoires.
L'enjeu de la situation 2 est la formulation de l'invariant opératoire de l'algorithme de calcul répétitif, dans l'écriture d'un message à une machine proche de celle de Babbage. Cette formulation est une condition nécessaire, à la conceptualisation de la notion de variable informatique.
L'enjeu de la situation 3 est l'écriture d'un programme itératif à la machine de Von Neumann. Cette écriture met jeu la formulation des conditions d'arrêt, l'opération d'affectation des variables informatiques ainsi que des éléments d'un langage évolué à la machine ordinateur.

Dans ces trois situations, l'élève a deux positions :

- Une position algorithmique : il travaille sur des algorithmes, solution du problème mathématique de tabulation
- Une position de programmation : il écrit des programmes à des machines successivement différentes.

Dans ces deux positions, il reçoit des informations et des rétroactions résultant de l'exécution de son programme sur des machines réelles ou partiellement fictives.
Nous schématisons ces deux positions dans la figure 2 ci-après.

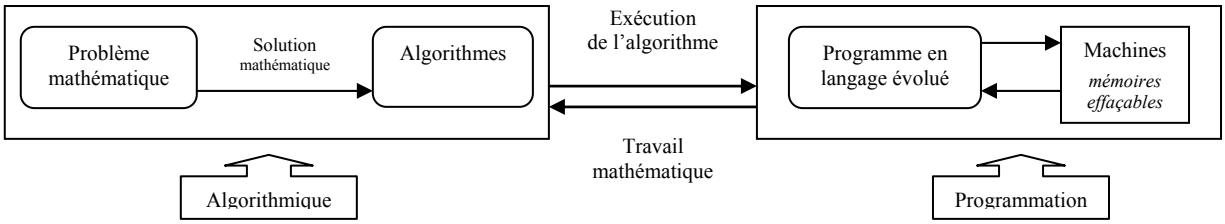


Figure 2. Les deux positions « algorithmique » et « programmation »

Les choix sur les nombres et la calculatrice Alpro

1. Les nombres dont on doit calculer les images appartiennent à D_j

L'écriture décimale des nombres est celle de l'affichage des résultats d'un calcul à l'aide d'une calculatrice ordinaire. Ce choix installe le calcul dans une problématique du calcul effectif à l'aide de la calculatrice tel qu'il existe dans les deux institutions.

2. Une calculatrice non programmable Alpro, machine de base de l'ingénierie didactique

Pour le besoin de notre recherche, nous avons conçu l'émulateur de la calculatrice Alpro sur le modèle des calculatrices existantes dans les deux institutions et ayant la capacité supplémentaire d'enregistrer l'historique des suites des touches appuyées lors d'un calcul.

Dans notre ingénierie, nous avons rendu indisponibles certaines touches de la calculatrice Alpro. Nous présentons ci-après l'interface graphique de cette version, appelée Alpro bloquée.

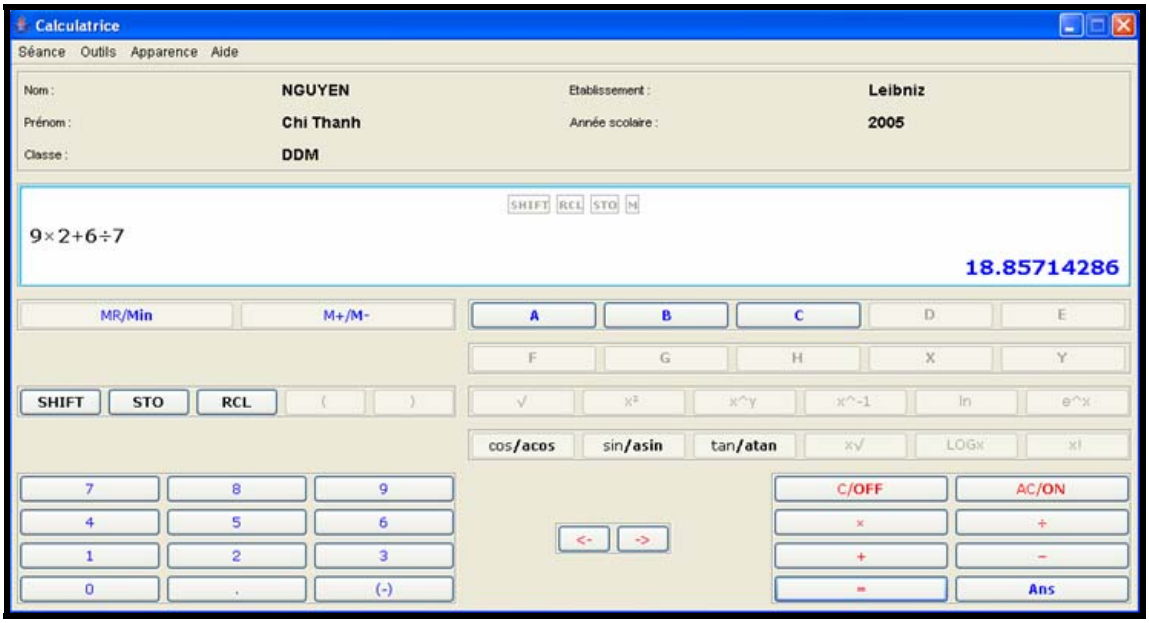


Figure 3. Aperçu global de l'émulateur Alpro, version bloquée

Les touches grisées sont indisponibles.
Précisons les principaux choix opérés sur la machine Alpro bloquée et les raisons de ces choix.

3. Les principales touches disponibles et indisponibles de la calculatrice Alpro bloquée

La mémoire Ans et la restriction à trois mémoires variables

L'analyse menée dans le chapitre C2 a mis en évidence l'existence dans les calculatrices non programmables existantes de deux types de touches mémoires visibles : touches mémoires variables A, B, C et touche mémoire du dernier résultat Ans. Par commodité, nous les appelons mémoires variables et mémoire Ans.

Ces deux types de mémoires ont des degrés de visibilité et d'effaçabilité différents. Elles sont destinées à recevoir des nombres pour un calcul effectif et sont « réutilisables » tout au long du calcul. Dans une mémoire variable, le stockage d'un nombre doit être intentionnel alors qu'un nombre est automatiquement stocké dans la mémoire Ans dès que la touche « = » (ou « Sto ») est pressée. Le nombre stocké intentionnellement dans une mémoire variable n'est effacé que quand l'opérateur décide d'y stocker un autre nombre ; alors qu'un nombre dans la mémoire Ans est effacé dès que la touche « = » (ou « Sto ») est pressée.

La calculatrice Alpro, à l'image des calculatrices non programmables, dispose des deux types de mémoires variable et Ans, mais à la différence de ces mêmes calculatrices, elle ne dispose que de trois mémoires variables A, B et C pour favoriser l'usage des mémoires variables et l'émergence de la notion de variable informatique. Les touches liées aux mémoires variables, « Sto » et « Rcl », sont disponibles ; on peut les taper directement sans devoir passer par la touche Shift.

Indisponibilité des touches parenthèses

La version utilisée rend indisponibles les touches parenthèses (mémoire invisible) pour favoriser ou rendre nécessaire l'usage des mémoires (variables ou Ans) lors de l'exécution ou de la programmation des calculs des images de nombres par une fonction.

Seules les touches des quatre opérations arithmétiques élémentaires sont disponibles

Nous avons choisi de ne conserver que les quatre opérations arithmétiques : « + », « - », « × », « ÷ ». Ce choix augmente, pour les multiplications en particulier, le coût des calculs puisque pour tout calcul, sans mémoire, de v^n il faut entrer le nombre v, n fois. L'augmentation du nombre de touches d'appuis accroît les risques d'erreurs et justifie l'usage des mémoires,
Par exemple, si on veut calculer v^4 avec v = 3,141759682, un calcul sans mémoire coûte 48 touches (3,141759682 × 3,141759682× 3,141759682 × 3,141759682 =), un calcul avec mémoire Ans coûte 20 touches (3,141759682 = Ans × Ans × Ans × Ans =)[1].

Accès aux informations sur les touches mémoires

L'émulateur donne accès à une information concernant les touches mémoires quand le curseur se palce à proximité de ces touches sur l'interface graphique.
Par exemple, le déplacement du curseur sur la touche « A » donne l'information suivante « **A** : Stocker une valeur dans la mémoire variable A. ». De même, le déplacement du curseur sur la touche « Ans » donne l'information « **Ans** : Rappeler le dernier résultat calculé. ».
Ces informations peuvent permettre d'amorcer l'instrumentation d'une touche mémoire variable ou mémoire Ans.

[1] Dans certaines calculatrices, Casio par exemple, on peut taper « 3,141759682 = Ans Ans Ans Ans = ». Ce qui donne 17 touches. Ce n'est pas le cas d'Alpro.

Nous allons maintenant entrer dans les situations construites de l'ingénierie didactiques en justifiant de plus près les choix issus du travail de conception et les conséquences prévues de ces choix sur les réponses et les stratégies (analyse *a priori*).

Puis nous examinons ce qui s'est passé dans les classes observées dans les deux pays (2nd et 1ère) dans une analyse *a posteriori*.

Dans la confrontation des analyses *a priori* et *a posteriori*, nous avons respecté le découpage en trois situations didactiques, qui n'est pas le découpage en séances de classe, ce dernier découpage obéissant à des contraintes institutionnelles de temps et d'organisation.

Une séance dans un lycée français dure 55 minutes alors qu'elle ne dure que 45 minutes dans un lycée vietnamien.
Dans la classe de 1e V, ceci nous a contraint à découper les trois situations didactiques en 4 séances. Mais l'organisation des séances a permis que deux séances de 45 minutes aient lieu un même jour en étant séparées par une pause de 10 minutes.

Le temps imparti à l'expérimentation de l'ingénierie didactique dans la classe de 1e F (165 minutes) a donc été plus court de 15 minutes par rapport à celui dans la classe de la 1e V (180 minutes).

Le tableau 1 présente les deux découpages (situations didactiques et séances) et les décalages pour la classe de 1e V.

<table>
<tr><th></th><th colspan="2">Situation didactique 1 (S1)</th><th colspan="2">Situation didactique 2 (S2)</th><th colspan="2">Situation didactique 3 (S3)</th></tr>
<tr><td>1e F</td><td colspan="2">Séance 1</td><td colspan="2">Séance 2</td><td colspan="2">Séance 3</td></tr>
<tr><td>1e V</td><td>Séance 1
- phases 1, 2&3</td><td>pause</td><td>Séance 2
- synthèse S1
- phases 1&2 (S2)</td><td>Séance 3
- phases 3, 4 (S2)
- phase 1 (S3)</td><td>pause</td><td>Séance 4
Fin S3</td></tr>
</table>

Tableau 1. Découpages en situations didactiques et en séances selon la classe 1e F ou 1e V

Chapitre 1

Première partie
Analyse *a priori* de la situation 1
Disponibilité et différenciation des mémoires variables et Ans

Introduction

La situation 1 est caractérisée par le triplet de valeurs des trois variables macro-didactiques de la situation fondamentale :

m = 1, **v ∈ D_j** ; machine Alpro bloquée.

L'enjeu de cette situation est d'organiser une première rencontre des élèves et de l'enseignant avec la machine calculatrice Alpro bloquée *(appelée Alpro par la suite*), d'observer (en particulier à l'aide des fichiers historiques d'Alpro) et de favoriser l'usage des mémoires variable et Ans.
Cette situation doit permettre de mettre à l'épreuve l'hypothèse de recherche suivante :

Dans EMS, la calculatrice fonctionne comme une machine arithmétique, c'est-à-dire une machine sans mémoire effaçable

Notre objectif, en jouant sur les valeurs des variables didactiques de la situation fondamentale, est ensuite de faire évoluer le rapport instrumental à la calculatrice, en particulier en permettant l'identification des touches mémoires effaçables et la différenciation de la mémoire du dernier résultat Ans d'avec les mémoires variables.
Elle doit aussi permettre la première rencontre avec un type de tâche rarement présent institutionnellement dans EMS (France et Viêt-nam) : « écrire à autrui un message de calcul pour que cet autrui l'exécute sur une machine »

I. Situation 1 – Calculs 1 et 2

La phase 1 de la situation 1 comprend deux questions résultant de choix sur les valeurs de variables didactiques de la situation fondamentale.

I.1. Énoncé des calculs 1 et 2 de la situation 1

Exercice 1 (10 minutes). Travail individuel avec la calculatrice Alpro ou la calculatrice personnelle. Avec la calculatrice personnelle, utiliser les mêmes touches autorisées qu'Alpro.

Question 1. Calculer $2x^2 + x + 1$ avec x = 3,141
Réponse 1 : ☐

Question 2. Calculer $8x^3 + 6x^2 - 3x - 1$ avec x = 3,141759682
Réponse 2 : ☐

Combien de fois en tout devez-vous appuyer sur les touches de la calculatrice Alpro pour réaliser le calcul complet de la question 2 ?
Nombre de fois : ☐

I.2. Variables didactiques du problème mathématique et saut informationnel entre le calcul 1 et le calcul 2

Nous avons joué essentiellement sur deux variables didactiques du problème mathématique, le degré de la fonction polynomiale et la valeur du nombre v.

- *Le degré n* prend d'abord la valeur 2 (question 1) pour rendre possible l'exécution des algorithmes de calcul à la main d'une image de la fonction (cf. chapitre C1) puis la valeur 3 pour augmenter le coût de ce calcul (question 3).
- *Le nombre v* appartient respectivement à D_3 (question 1) et D_9 (question 2). L'appartenance à D_3 est choisie pour permettre d'observer les pratiques routinières de calcul chez les élèves. Le choix d'un nombre appartenant à D_9 augmente considérablement le coût du calcul à la main, et devrait favoriser l'usage des mémoires, si elles sont disponibles.

Un choix secondaire est celui des coefficients de la fonction polynomiale de la question 2 tous différents de 0 et 1 afin d'accroître le coût du calcul.

Le passage entre les deux calculs (question 1 et question 2) organise un saut informationnel visant à changer le rapport aux touches mémoires de la calculatrice :

> Le saut informationnel consiste, après avoir trouvé une situation fondamentale faisant « fonctionner » une notion, à choisir d'abord les valeurs de ses variables de telle manière que les connaissances antérieures des élèves permettent d'élaborer des stratégies efficaces … puis, sans modifier les règles du jeu, à changer les valeurs des variables de façon à rendre beaucoup plus grande la complexité de la tâche à accomplir. De nouvelles stratégies doivent être établies qui demandent la construction de nouvelles connaissances. (Brousseau 1986, p. 23)

En effet, les choix portant sur la valeur du degré n et sur le type de nombre décimal v dont on doit calculer l'image, en augmentant le coût du calcul, changent sa complexité dans le passage de la question 1 à la question 2 ; la stratégie optimale dans la question 2 est alors une stratégie utilisant le stockage du nombre v dans une mémoire pour permettre le calcul le plus économique.
La demande du nombre de touches appuyées permet :

- d'ajouter un enjeu à ce deuxième calcul : celui d'utiliser le moins de touches possibles.
- de rendre formulable le coût des stratégies de calcul
- de préparer la comparaison publique des algorithmes de calcul du point de vue de ce coût dans une phase d'institutionnalisation des mémoires.

L'analyse *a priori* des différentes stratégies algorithmiques de calcul possibles et leur comparaison en termes d'opérations et de mémoire dans chacun des deux calculs va nous permettre de mettre en évidence ce saut informationnel.

I.3. Stratégies possibles

Nous nous référons pour cette analyse à l'étude menée dans le chapitre C1.
Dans ce chapitre, nous avons identifié trois algorithmes de calculs pour le calcul d'une image d'une fonction polynomiale. Nous rappelons ci-après ces algorithmes de calcul.

Ad : Calcul utilisant l'algorithme de calcul direct

- calculer chacun des termes a_nx^n, $a_{n-1}x^{n-1}$, …a_1x, a_o en remplaçant x par v ;
- additionner les résultats.

Ap : Calcul utilisant l'algorithme de calcul direct sur les puissances

- former une table [1..n] pour contenir les valeurs de x, x^2, x^3…x^{n-1}, x^n quand on remplace x par v. Un terme est calculé en multipliant le terme qui le précède par v.
- multiplier chaque terme par son coefficient ;
- additionner tous les produits.

Ah : Calcul utilisant l'algorithme de Hörner ;

$f(x) = ((\ldots(a_nx + a_{n-1})x + \ldots)x + a_o$

Le calcul de la valeur de f(x) en la valeur v débute par le calcul de la valeur de l'expression la plus intérieure : $a_nv + a_{n-1}$, puis on multiplie le résultat par v, on ajoute a_{n-1} au produit, ainsi de suite. Le dernier résultat obtenu ne doit pas être mémorisé, il est utilisé directement dans la suite du calcul.

Seul le premier algorithme est attendu, le second étant peu probable, le troisième étant improbable puisque non enseigné. L'analyse a priori prend en considération les trois algorithmes, mais met en avant dans les conclusions l'algorithme de calcul direct Ad.

Par commodité, nous utilisons les symboles × et + pour désigner respectivement la multiplication et l'addition ; l'entrée d'un nombre dans la calculatrice, par exemple 3,141 est notée : 3,141 ; les touches mémoires variables sont appelées : mémoires variables, la touche mémoire Ans : mémoire Ans.

I.3.1. Calcul 1 (Question 1)

Pour ce premier calcul, nous avons choisi :

- $v = 3{,}141 \in D_3$ mentalement mémorisable,
- la fonction f telle que $f(x) = 2x^2 + x + 1$ avec $f(3{,}141) = 23{,}872762 \in D_6$.

Tout d'abord, nous faisons l'hypothèse que le choix d'un nombre décimal et la présence d'une calculatrice mettent l'élève dans le contrat du calcul effectif, c'est-à-dire instrumenté par la calculatrice. C'est pour cela que nous ne considérerons dans cette analyse *a priori* que ce type de calcul.

Stratégies de calcul avec la machine arithmétique (c'est-à-dire sans mémoire effaçable)
Les trois algorithmes de calcul Ad, Ap et Ah auparavant cités sont possibles sans utiliser de mémoires effaçables (cf. chapitre C1).
Le tableau suivant présente le coût du calcul instrumenté par la calculatrice de ces différents algorithmes en termes des nombres d'opérations et d'appuis sur les touches de la calculatrice, machine arithmétique :

Stratégies / Composantes du coût	Nd	Np	Nh
× et +	2 × ; 2 +	2 × ; 2 +	2 × ; 2 +
touches	22	22	19

Tableau 1. Coût du premier calcul suivant les trois stratégies à l'aide d'une machine arithmétique

La formule de la fonction f a été choisie de façon à rendre les deux stratégies Nd et Np équivalentes en ce qui concerne le coût du calcul instrumenté (nombre d'opérations ; nombre d'appuis de touches).
Les algorithmes de calcul Np et Nh obligent à stocker un unique résultat intermédiaire (RI) :

- $x \times x$ pour Np ;
- $2 \times x + 1$ pour Nh.

Du fait qu'il y a un unique RI et que ce résultat intervient dans la suite du calcul, le calcul peut se faire de deux façons, par exemple pour l'algorithme Nh :

- soit en utilisant la mémoire papier (écriture de RI sur le papier)), puis en multipliant RI par x et en ajoutant 1 à ce dernier résultat ;
- soit en utilisant implicitement la touche mémoire « Ans » comme suit : [2] [×] [3,141] [+] [1] [=] [×] [3,141] [+] [1] [=].

La touche mémoire Ans est utilisée sans que le symbole « Ans » apparaisse dans la suite des appuis des touches.
La stratégie Nh est la moins coûteuse en termes de calcul instrumenté, mais exige l'usage d'une mémoire (explicitement la mémoire papier ou implicitement la mémoire Ans).

Stratégies de calcul avec une machine à mémoire effaçable
Nous pouvons envisager les trois algorithmes de calcul Ad, Ap et Ah, avec une calculatrice à mémoire effaçable, mémoires Ans ou mémoire variable.

• Md : Calcul utilisant l'algorithme de calcul direct

- Md_v[1] mémoires variables A, B et C :

  [3,141] [Sto] [A] [2] [×] [A] [×] [A] [+] [A] [+] [1] [=] ;

- Md_a mémoire Ans :

  [3,141] [=] [2] [×] [Ans] [×] [Ans] [+] [Ans] [+] [1] [=].

• Mp : Calcul utilisant l'algorithme de calcul direct sur les puissances

- Mp_v mémoires variables A, B, C :

  [3,141] [Sto] [A] [A] [×] [A] [Sto] [B] [2] [×] [B] [+] [A] [+] [1] [=] ;

- Mp_a mémoire Ans : comme il y a deux stockages successifs nécessaires, l'usage de la mémoire Ans n'est pas possible.

• Mh : Calcul utilisant l'algorithme de Hörner

- Mh_v mémoires variables A, B, C :

  [3,141] [Sto] [A] [2] [×] [A] [+] [1] [Sto] [B] [B] [×] [A] [+] [1] [=] ;

- Mh_a mémoire Ans : l'usage de la mémoire Ans est impossible pour les mêmes raisons données précédemment.
- Mh_{av} mémoire A, B, C et Ans : Mh peut mobiliser une mémoire variable et la mémoire Ans au cours du calcul

  [3,141] [Sto] [A] [2] [×] [A] [+] [1] [=] [×] [A] [+] [1] [=] ;

Le tableau 2 présente le coût du calcul instrumenté par une calculatrice à mémoire effaçable de ces différents algorithmes en termes des nombres d'opérations et d'appuis sur les touches de la calculatrice.

[1] Nous utilisons les notations, par exemple Md_v ou Md_a pour désigner les stratégies qui utilisent seulement la mémoire variable ou seulement la mémoire Ans.

Stratégies / Composantes du coût	Md		Mp		Mh		
	Md_a	**Md_v**	**Mp_a**	**Mp_v**	**Mh_a**	**Mh_v**	**Mh_{av}**
× et +	2× ; 2 +	2× ; 2 +	Impossible	2× ; 2 +	Impossible	2× ; 2 +	2× ; 2 +
mémoires	Ans	A		A, B		A, B	Ans et A
touches	16	17		20		20	18

Tableau 2. Coût du premier calcul suivant les trois stratégies à l'aide d'une machine à mémoires effaçables

La comparaison des coûts du calcul instrumenté par une machine arithmétique (tableau 1) à ceux par une machine à mémoire effaçable (tableau 2) montre que l'usage des mémoires apporte un gain quasi nul au calcul.
Si les mémoires ne sont pas utilisées, cela ne signifie pas nécessairement que les élèves ne les connaissent pas. Cela signifie seulement que l'usage des mémoires ne fait pas partie de leurs pratiques routinières avec la calculatrice.

I.3.2. Calcul 2 (Question 2)

Pour ce second calcul, nous avons choisi :

- le nombre v = 3,141759682 $\in D_9$
- la fonction polynomiale f telle que $f(x) = 8x^3 + 6x^2 - 3x - 1$, avec f(3,141759682) = 296,888424 : résultat affiché sur Alpro (le résultat avec les chiffres de réserve est : 296,88842400055)

Il est possible de transformer $f(x) = 8x^3 + 6x^2 - 3x - 1$ en un produit $(x + 1)(4x + 1)(2x - 1)$ ou en une formule canonique de Cardan $8X^3 - 4,5X$ avec $X = x + 0,25$.
Ces formules diminuent peu le nombre d'appuis de touches puisque Alpro ne dispose pas de touches parenthèses. Mais l'apparition de ces formules est un observable de la force du contrat de l'algèbre contre celui du calcul effectif. On peut s'y attendre au Viêt-nam où l'algèbre domine les pratiques de calcul.

Stratégies de calcul avec la machine arithmétique (c'est-à-dire sans mémoire effaçable)
Le tableau 3 suivant résume le coût de calcul en termes des nombres d'opérations et du nombre d'appuis de touches selon les trois algorithmes Ad, Ap et Ah et ce sans utiliser les mémoires d'Alpro :

Stratégies / Composantes du coût	Nd	Np	Nh
× et +	6 × ; 3 +	5 × ; 3 +	3 × ; 3 -
Touches	80	134	68

Tableau 3. Coût du deuxième calcul suivant les trois stratégies à l'aide d'une machine arithmétique

Pour les deux stratégies Np et Nh, il faut stocker en mémoire les résultats intermédiaires (RI) :

- pour Np : les puissances x, x^2 et x^3 ;
- pour Nh : les images du nombre v par les fonction affines imbriquées dans le produit $((8x + 6)x - 3)x - 1$.

L'utilisation de la mémoire papier est un observable de la non utilisation des mémoires de la calculatrice
La stratégie Np exige le plus grand nombre d'appuis de touches car il faut retaper des RI et le nombre v sur l'écran. Elle est la plus coûteuse.
La stratégie Nh exige à la fois le plus petit nombre d'appuis de touches et le plus petit nombre d'opérations, mais il faut noter deux résultats intermédiaires sur le papier. Ces recopies des RIs sont des sources d'erreurs de calcul.

La stratégie Nd n'exige aucune recopie de RI et demande un nombre d'appuis de touches comparable aux autres : elle n'est donc pas bloquée mais le nombre élevé d'appuis de touches favorise la recherche de l'usage des mémoires *à condition que ces touches soient connues et disponibles.*

Stratégies de calcul avec une machine à mémoire effaçable
Nous pouvons envisager les trois algorithmes de calcul Ad, Ap et Ah, avec une calculatrice à mémoire effaçable, mémoires Ans ou mémoire variable.

• Md : Calcul utilisant l'algorithme de calcul direct.
- Md_v mémoires variables A, B, C :
  3,141759682 Sto A 8 × A × A × A + 6 × A × A − 3 × A − 1 = ;
- Md_a mémoire Ans :
  3,141759682 = 8 × Ans × Ans × Ans + 6 × Ans × Ans − 3 × Ans − 1 =.

• Mp : Calcul utilisant l'algorithme de calcul direct sur les puissances.
- Mp_v mémoires A, B, C :
  3,141759682 Sto A A × A Sto B B × A Sto C 8 × C + 6 × B − 3 × A − 1 =.

Cette stratégie mobilise donc les trois mémoires variables A, B, C.
- Mp_a mémoire Ans : le stockage devant se faire à plusieurs reprises durant le calcul, l'usage de la mémoire Ans est impossible.

• Mh : Calcul utilisant l'algorithme de Hörner ;
- Mh_v mémoires variables A, B, C :
  3,141759682 Sto A 8 × A + 6 Sto B B × A − 3 Sto C C × A − 1 =.

Cette stratégie mobilise donc les trois mémoires variables A, B, C.
- Mh_a mémoire Ans : il faut utiliser trois stockages donc il est impossible d'utiliser uniquement la mémoire Ans.
- Mh_{av} mémoire A, B, C et Ans : Par contre, il est possible de combiner Mh_v et Mh_a en utilisant une mémoire variable et la mémoire Ans :
  3,141759682 Sto A ; 8 × A + 6 = × A − 3 = × A − 1 =.

Le tableau 4 présente le coût du calcul instrumenté par une calculatrice à mémoire effaçable du deuxième calcul avec les différentes stratégies en termes des nombres d'opérations et d'appuis sur les touches de la calculatrice, machine à mémoires effaçables.

Stratégies / Composantes du coût	Md		Mp		Mh		
	Md_a	**Md_v**	**Mp_a**	**Mp_v**	**Mh_a**	**Mh_v**	**Mh_{av}**
× et +	6× ; 3+	6× ; 3+	Impossible	5× ; 3+	Impossible	3× ; 3+	3× ; 3+
Mémoires	Ans	A		A,B,C		A,B,C	A, Ans
Touches	**32**	**33**		**37**		**33**	**29**

Tableau 4. Coût du deuxième calcul suivant les trois stratégies à l'aide d'une machine à mémoires effaçables

Stratégies de calcul sans ou avec mémoire effaçable
Le coût de calcul élevé dans les stratégies sans mémoire (tableau 3), notamment en termes de nombre d'appuis de touches (de 68 à 134) favorise l'usage des mémoires dans le calcul, *si elles sont disponibles*. Comme dans le calcul 1, la stratégie Mp est la plus coûteuse en termes de nombre d'appuis des touches (134).

Le stockage de v = 3,141759682 dans une mémoire réduit le nombre d'appuis des touches nécessaires à l'entrée du nombre.
Dans le cas de l'algorithme de calcul direct, la mémoire Ans et la mémoire variable ont le même statut et sont donc en concurrence. Il donne donc la possibilité d'observer la disponibilité de l'une ou l'autre des deux mémoires variables

A la fin du calcul 2, l'enseignant demande à un élève ayant obtenu un résultat correct et ayant tapé le plus petit nombre de touches de rendre public son calcul avec Alpro, l'objectif étant de rendre public l'usage des mémoires Ans ou variable.
On suppose donc dans l'analyse *a priori* du calcul 3 que les touches mémoires sont « connues » sans pour autant être instrumentées.

II. Situation 1 – Calcul 3 (exercice 2)

Ce calcul a pour objectif l'écriture d'un message de calcul à un autre binôme et sa validation par son exécution sur Alpro.
Ce message doit être le plus court possible et ne contenir qu'une suite de touches d'Alpro. On peut considérer que ce message est un programme informatique dans le langage évolué destiné à Alpro.
Nous voulons ainsi :

- rendre nécessaire, en raison de l'absence des mémoires parenthèses, l'usage des mémoires et la distinction entre deux types mémoires, la mémoire Ans comme mémoire du dernier résultat et la mémoire variable comme mémoire de stockage volontaire d'un nombre ;
- organiser la première rencontre avec le type de tâche : « écrire à autrui un message de calcul pour qu'il l'exécute sur une machine ».

II.1. Énoncé du calcul de la situation 1 – Calcul 3

Exercice 2 (25 minutes). Travail en binôme avec la calculatrice Alpro
Feuille de consigne du binôme :

Phase 1 ***(10 minutes)*** *: Vous disposez d'un stylo et de la calculatrice Alpro*
Sur la feuille navette (phase 1), écrivez un message le plus court possible à un autre binôme qui ne possède ni papier ni stylo. Il dispose seulement de la même calculatrice que vous. Dans le message n'indiquez que les touches de la calculatrice. En suivant vos instructions, ce binôme doit obtenir la valeur numérique de z à l'écran de la calculatrice.

Variante 1 : $z = y^2 - 7y + 9$ où $y = \dfrac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$ avec x = 1,257

Variante 2 : $z = y^2 - 5y + 7$ où $y = \dfrac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$ avec x = 1,254

Phase 2 ***(5 minutes)*** *: Rangez votre stylo. Vous n'avez plus que la calculatrice Alpro*
Exécutez, avec la calculatrice Alpro, toutes les instructions du message reçu, tant que c'est possible.
Reprenez alors votre stylo pour recopier sur la feuille navette (phase 2) le dernier écran de la calculatrice.

Phase 3 ***(10 minutes)*** *: Vous disposez d'un stylo et de la calculatrice Alpro*
Sur votre fiche navette qui vient de vous être rendue, proposez si nécessaire, un nouveau message.

L'un des binômes sera choisi au hasard et les instructions de son message seront exécutées publiquement sur la calculatrice Alpro.

Les variantes 1 et 2 du calcul 3 ont été rendues nécessaires par la situation de communication : en effet celle-ci est basée sur l'échange de messages entre deux binômes (phase 2) qui ne doivent pas avoir le même calcul à résoudre.

Ces variantes ont été construites pour être équivalentes du point de vue de l'écriture du message comme nous le montrerons par la suite.

II.2. Variables didactiques du problème mathématique et saut informationnel entre le calcul 2 et le calcul 3 de la situation 1

Le calcul de l'image d'une fonction est le problème mathématique central de notre ingénierie. Ici il concerne le calcul de l'image d'une fonction composée f◦g.
Dans un tel calcul et en l'absence de parenthèses, il faut calculer d'abord l'image d'un nombre v par g, mémoriser le nombre g(v), puis calculer l'image de g(v) par f.
Pour rendre nécessaire l'usage des mémoires, nous avons choisi la fonction g de telle sorte que le remplacement direct de g(v) dans l'expression de f(x) ne soit pas possible ou trop coûteux. C'est pour cela que nous avons choisi g tel que $g(x)=\frac{N(x)}{M(x)}$ où N et M sont des fonctions polynomiales.
$g(x)=\frac{N(x)}{M(x)}$ n'est pas réductible et la simplification de f◦g(x), possible algébriquement[1] est très improbable dans le contrat du calcul effectif et de toute façon très coûteuse voire impossible dans le temps imparti.

Contrairement au calcul 2 où c'est l'appartenance de v à D_9 qui favorise l'usage des mémoires dans le calcul 3 c'est le fait que f soit une fonction composée f◦g combiné au blocage des parenthèses qui exige l'usage des mémoires.
Nous nous sommes contentés de prendre un nombre appartenant à D_3 ; mais nous l'avons choisi de telle façon que l'image de v par g soit un nombre décimal ayant un zéro comme troisième chiffre après la virgule (1,42030961 ou 1,41051891). Y aura-t-il troncature du résultat pour éviter l'usage d'une mémoire ?

Les calculs 2 et 3 se distinguent par l'usage des mémoires.
Dans le calcul 2, l'usage des mémoires est favorisé par la recherche d'un nombre minimum d'appuis des touches de la calculatrice, mais cet usage n'est pas obligatoire. De plus, dans le cas de l'algorithme de calcul direct (le plus probable), les mémoires variables et la mémoire Ans ont le même statut du fait qu'il y a un seul appui sur la touche « = » durant le calcul.
Dans le calcul 3, le calcul de l'image du nombre v par une fonction composée et le blocage des parenthèses, rendent obligatoire l'usage des mémoires et la différentiation des deux types de mémoires du fait qu'il y a plus d'un appui sur la touche « = » durant le calcul.

II.3. Stratégies possibles

Nous donnons d'abord dans le tableau 5, pour les deux calculs, les images z = f◦g(v) = f(y) affichées sur l'écran de la calculatrice. Ce tableau montre l'équivalence des deux variantes pour le calcul et donc pour l'écriture des messages.

---

[1] $f\circ g(x)=0{,}0164365549\ x^4+0{,}281511826\ x^3-0{,}9352391177\ x^2-0{,}6813344014\ x+\frac{-3{,}552439741}{x-0{,}8563589744}$
$+\frac{0{,}25607677}{(x-0{,}8563589744\ )^2}+10{,}62136543$

x = 1,254	$y = \frac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$	$z = y^2 - 5y + 7$
Valeurs affichées	N =17,75387013 M = 12,5 y = 1,42030961	1,915731338
Variante 1		

x = 1,257	$y = \frac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$	$z = y^2 - 7y + 9$
Valeurs affichées	N = 17,63148637 M = 12,5 y = 1,41051891	1,115931226
Variante 2		

Tableau 5. Les résultats des calculs des images f∘g(v)

L'usage des mémoires est indispensable dans ce calcul.

Stratégies de calcul avec la machine arithmétique

Cependant nous pouvons prévoir que l'indisponibilité des mémoires variables et le blocage des mémoires parenthèses conduisent à des subterfuges dans l'écriture du message.

Pour le calcul de z = f∘g(v), les stratégies de calcul sans mémoire ne sont possibles que si les résultats intermédiaires sont notés sur papier ou mémoriser mentalement. Nous avons bloqué cette possibilité en précisant que « *l'autre binôme ne possède ni papier ni stylo* ».

Une autre manière d'éviter l'usage des mémoires est de noter les résultats intermédiaires, notamment M(v) et N(v) sur le message lui-même, autrement dit la mémoire papier est sollicitée à la place des mémoires de la machine.

Ce subterfuge sera pour nous un observable de l'indisponibilité des mémoires et donc de la résistance du rapport à la calculatrice comme machine arithmétique.

Stratégies de calcul avec la machine à mémoire effaçable

Les stratégies de calcul d'une image par une fonction polynomiale, N(v) et z = f∘g(v), ont déjà été présentées dans l'analyse *a priori* de la phase 1.

En tenant compte du temps imparti, du blocage des parenthèses et de la nature de la fonction g, nous considérons que l'algorithme de calcul direct est le seul probable.

Il donne lieu à trois stratégies possibles M_v , M_{vm} et M_{av} selon l'usage des mémoires.

- M_v mémoires variables,
- M_{vm} mémoire variable mathématique
- M_{av} mémoires variable et mémoire Ans

La stratégie de calcul commence de manière identique:

• stockage du nombre v dans la mémoire A ;
• calcul du numérateur puis stockage du nombre N(v) dans la mémoire B ;
• calcul du dénominateur puis stockage du nombre M(v) dans la mémoire C ou dans la mémoire Ans ;

1. *Si stockage du nombre M(v) dans la mémoire C*

• calcul de B ÷ C : les trois mémoires variables sont utilisées.

La limitation à trois mémoires variables dans la version Alpro bloquée (contrairement à la plupart des calculatrices non programmables actuelles) a été choisie pour faire rencontrer la question instrumentale **:**

Où stocker un résultat quand toutes les mémoires disponibles ont déjà été utilisées ?

Trois réponses sont possibles pour répondre à cette question :

Réponse 1 (A, B, C, A) : M_v Réutiliser l'une des mémoires variables. Cette mémoire prend alors des caractéristiques de mémoire effaçable. La suite de touches est la suivante :

1,254 Sto A 2 × A × A × A + 15 × A − 5 Sto B 32,1 × A - 26,7184 Sto C B ÷ C Sto A A × A − 5 × A + 7 = ;

Réponse 2 (A, B, C, B÷C) : M_{vm} Les lettres désignant les mémoires et contenant des valeurs différentes sont utilisées dans le message comme des variables mathématiques :
1,254 Sto A 2 × A × A × A + 15 × A − 5 Sto B 32,1 × A - 26,7184 Sto C B ÷ C × B ÷ C − 5 × B ÷ C + 7 = ;

Réponse 3 (A, B, C, Ans) : M_{av} Le calcul de B ÷ C est suivi de l'appui sur la touche = pour pouvoir utiliser la mémoire Ans. La suite de touches est la suivante :
1,254 Sto A 2 × A × A × A + 15 × A − 5 Sto B 32,1 × A - 26,7184 Sto C B ÷ C = Ans × Ans − 5 × Ans + 7 =.

2. Si stockage du nombre M(v) dans la mémoire Ans

• Calcul de B ÷ Ans : ce dernier résultat peut être stocké

Réponse 4 (A, B, Ans, Ans) : de nouveau, dans la mémoire Ans puis calcul de z. La suite de touches est la suivante :
1,254 Sto A 2 × A × A × A + 15 × A − 5 Sto B 32,1 × A - 26,7184 = B ÷ Ans = × Ans − 5 × Ans + 7 = ;

Réponse 5 (A, B, Ans, C) dans la mémoire variable C (A et B étant déjà utilisées) puis calcul de z. La suite de touches est la suivante :
1,254 Sto A 2 × A × A × A + 15 × A − 5 Sto B 32,1 × A - 26,7184 = B ÷ Ans = Sto C C × C − 5 × C + 7 =.

A la différence des stratégies 3 et 5 (une seule utilisation de Ans), la stratégie 4 (A, B, Ans, Ans) utilise Ans deux fois dans le calcul. Dans les réponses 2 (A, B, C, B÷C), 3 (A, B,C,Ans) et 5 (A, B, Ans, C), les mémoires variable ou Ans ne contiennent qu'un seul nombre durant tout le calcul. Dans ce sens, ces mémoires fonctionnent comme des variables mathématiques.

Rappelons que, dans le calcul 3, la mémoire Ans ne peut pas (théoriquement) être utilisée toute seule, c'est-à-dire sans mémoire variable.

Nous rassemblons dans une même stratégie de calcul M_{av} les trois calculs (Réponses 3, 4 et 5), instrumentés par Alpro et utilisant les deux types de mémoires, Ans et variable. Le coût du calcul des trois stratégies M_v , M_{vm} et M_{av} est résumé dans le tableau 6 ci-après :

Stratégies / Composantes du coût	**M_a**	**M_v**	**M_{vm}**	**M_{av}**
× et +	Impossible	7 ×, 5 +	7 ×, 3 ÷, 5 +	7 ×, 5 +
Mémoires		A, B, C	A, B, C	A,B,C et Ans
Touches		54	55	52-54

Tableau 6. Coût du calcul de f∘g(v) suivant les stratégies à l'aide d'une machine à mémoires effaçables

II.4. Conclusion de l'analyse *a priori* du calcul 3

Le calcul 3 de la situation 1 permet de bloquer l'usage de la mémoire Ans en tant que seule mémoire possible dans le calcul et par conséquent de distinguer les deux types de mémoires : variables et Ans.

Cependant les valeurs contenues dans les mémoires variables peuvent ne pas changer durant le calcul. *Deux stratégies possibles évitent l'usage d'une même mémoire pour stocker intentionnellement deux nombres différents*, l'une M_{vm} en considérant les mémoires variables comme des variables mathématiques, l'autre M_{av} en utilisant conjointement mémoires variables et mémoire du dernier résultat Ans (sauf pour la variante (A, B, Ans, Ans) où la mémoire Ans reçoit deux nombres différents). Ces stratégies donnent aux mémoires variables et Ans un même statut, celui des variables mathématiques : chaque mémoire reçoit un seul nombre durant le calcul.

Seule la stratégie M_v s'appuie sur la propriété d'effaçabilité de la mémoire variable. Or la notion de variable en informatique, mémoire effaçable désignée, prend pleinement sa signification lorsque son contenu reçoit successivement et intentionnellement des valeurs différentes. De ce point de vue, la stratégie M_v est un observable d'un premier fonctionnement de la mémoire variable comme une variable informatique dans un calcul non répétitif.

La situation 2 aura pour objectif de faire évoluer le statut de la mémoire variable afin de donner pleinement son sens à la notion de variable informatique.

III. Conclusion de l'analyse *a priori* de la situation 1

Nous résumons dans le tableau ci-après les valeurs choisies pour les variables didactiques, et les mémoires utilisées dans les algorithmes optimaux de calcul direct.

Variables	**Degré n**	**Nombre v**	**Mémoires des stratégies optimales**	**Ce que l'on veut observer**
Calcul 1	2	$v \in D_3$	Sans mémoire ou Ans ou A, B, C	Pratiques routinières
Calcul 2	3	$v \in D_9$	Ans ou A, B, C	Disponibilité des mémoires
Calcul 3	3, 2, 1	$v \in D_3$; $g(v) \in D_8$	A, B, C ou A, B, C et Ans	Fonctionnement des mémoires

Tableau 7. Variables didactiques et valeurs choisies dans la situation 1

Ce tableau résume les conditions d'observation des deux sauts informationnels organisés, l'enjeu d'enseignement étant de rendre disponible les mémoires variables et Ans et de les différencier.

Le saut informationnel opéré entre les deux premiers calculs favorise théoriquement dans le calcul 2 l'usage de deux types de mémoires qui ont, pour le calcul le même statut : stocker un nombre pour réutiliser ce nombre dans un calcul.
Le saut informationnel opéré entre le calcul 2 et le calcul 3 rend théoriquement obligatoire, le stockage intentionnel d'un nombre dans des mémoires variables. Le calcul 3 et la limitation du nombre des mémoires variables à trois peuvent permettre, sans la rendre nécessaire, l'évolution des mémoires variables vers un premier fonctionnement en tant que variable informatique : un observable étant la réutilisation d'une même mémoire variable.

Un certain nombre d'observables permettent de rendre compte du rapport à la calculatrice comme machine arithmétique et de la résistance de ce rapport aux tentatives de déstabilisation que sont les deux sauts informationnels et la phase d'institutionnalisation à la fin de la phase 1.
Ces observables sont donnés dans le tableau 8 ci-après.

	Calcul 1	Calcul 2	Calcul 3
Ce que l'on veut observer	Pratiques routinières	Disponibilités des mémoires	Fonctionnement des mémoires
Observables	Historiques	Historiques Nombre d'appuis des touches	Historiques RI écrit dans le message Troncature de g(v)

Tableau 8. Observables possibles dans la situation 1

Que s'est-il passé ?
Nous allons maintenant analyser les réalisations de la situation 1 dans EMS en France et au Viêt-nam pour répondre à cette question.

Chapitre 1

Deuxième partie
Analyse *a posteriori* de la situation 1

I. Les conditions de l'expérimentation et du recueil des données

L'expérimentation de la situation 1 (phases 1 et 2) a été conduite dans cinq classes :

- Deux classes de 2nde (14 mai 2004, Lycée Moyen Grésivaudan de Villard-Bonnot, région grenobloise) et une classe de 1er S (10 janvier 2005, Lycée français Alexandre Yersin, Hanoi)
- Une classe 10 (3 janvier 2005, Lycée d'application de l'Université Pédagogique d'Ho Chi Minh) et une classe 11 (7 janvier 2005, Lycée Chu Van An, Hanoi)[1]

Pour simplifier, nous notons par la suite ces classes 2nd F1, 2nd F2 et 1er F pour les classes de Lycées Français ; 2nd V, 1er V pour les classes de Lycées vietnamiens.
L'effectif de chaque classe est donné dans le tableau 1 suivant :

Classes	2nd F1	2nd F2	2nd V	1er F	1er V	Total France	Total Viêt-nam	Total
Effectifs	33	33	42	20	18	86	60	146

Tableau 1. Effectifs des classes dans l'expérimentation

Nous avons expérimenté l'ensemble de l'ingénierie didactique dans les deux classes de 1er F et de 1er V, indiquées en grisé dans le tableau 1 : Alain Birebent est l'enseignant de la classe 1er F et Nguyen Chi Thanh est celui de la classe de 1er V. Annie Bessot est observatrice dans les deux classes.

Les analyses s'appuient sur les données suivantes :

- Les notes des observateurs[2] ;
- Les fiches de réponses des élèves et leurs brouillons ;
- Les enregistrements audios des élèves : 16 élèves de seconde en France ; deux élèves dans la classe de 1er S du Lycée français Alexandre Yersin et deux élèves dans la classe 11 du Lycée Chu Van An.
- L'enregistrement vidéo des enseignants : dans toutes les classes sauf dans le Lycée d'application de l'Université Pédagogique d'Ho Chi Minh ;
- Les fichiers historiques : dans toutes les classes pour les élèves travaillant avec Alpro. Un incident technique n'a permis de recueillir que 10 fichiers historiques de la classe 10 du Lycée d'application de l'Université Pédagogique d'Ho Chi Minh.

Pour les trois classes 1er F, 2nd V et 1er V, faute d'un nombre suffisant d'ordinateurs, la moitié des élèves ont du travailler avec leur calculatrice personnelle pour les calculs 1 et 2, avec la

[1] Nous remercions ici Christophe Daudin enseignant des classes de 2nd F1, 2nd F2, Olivier Tawin enseignant de la classe 1er F, Le Thanh Thai enseignant de la classe de 2nd V, Nguyen Thi Thuy enseignante de la classe de 1er V et leurs élèves de nous avoir accueillis.
[2] Annie Bessot pour toutes les classes et Alain Birebent pour toutes sauf celle où il enseigne.

consigne : « *Avec la calculatrice personnelle, utiliser les mêmes touches autorisées qu'Alpro* ». Les touches interdites ont été notées au tableau par l'enseignant.
L'autre moitié a travaillé avec Alpro et nous n'avons pu recueillir les fichiers des historiques de touches appuyées que pour ces élèves.
La majorité des calculatrices utilisées par les élèves de 1er V sont de marques Casio : Fx500MS ou Fx500A non programmables. Toutes les calculatrices utilisées par les élèves de 1er F sont programmables de marques Casio Graph ou TI.
Les classes 2nd F1, 2nd F2 ont été partagées en deux groupes pour pouvoir travailler sur Alpro en salle d'ordinateurs du Lycée du Moyen Grésivaudan (Isère-France).

II. Situation 1 – calculs 1 et 2

II.1. Calcul 1

Nous rappelons le calcul 1 : calculer $2x^2 + x + 1$ avec $x = 3{,}141$.
Le résultat attendu est 23,872762 (valeur exacte du nombre qui est aussi la valeur affichée).
Nous analysons d'abord les résultats donnés par les élèves dans leurs fiches de réponses, puis les procédures de calcul à partir de l'analyse des fichiers historiques et des brouillons.

II.1.1. Quel résultat du calcul 1 est donné dans les productions écrites ?

Comme nous l'avions prévu, ce calcul n'est pas problématique pour des élèves dans les deux institutions : tous les élèves ont répondu à la question et seulement quatre réponses (sur 146 réponses, soit 2,73%) sont incorrectes.
Toutes les erreurs sont celles d'élèves de l'institution française : 3 en 2nd F1 et 1 en 1er F. À partir des fichiers historiques, nous n'avons pu reconstitué que le calcul des 3 élèves de 2nd F1.

Elèves	Calcul instrumenté	Règles de calcul expliquant l'erreur
2 (2nd F1)	$2 \times 3{,}141 + 3{,}141 + 1$	$2x^2 + x + 1 \rightarrow 2 \times x + x + 1 = 3x + 1$
1 (2nd F1)	$2 \times 3{,}141$ = [6,282 à l'écran] $\times$ 6,282 = Ans + 3,141 = Ans + 1	$2x^2 + x + 1 \rightarrow (2 \times x) \times (2 \times x) + x + 1 = 4x^2 + x + 1$

Tous les élèves vietnamiens ont donné un résultat avec tous les chiffres affichés à l'écran de la calculatrice.
Par contre, 13 élèves de l'institution française - 11 en 2nde et 2 en 1er - soit 15% des élèves français, ont donné des valeurs approchées, tronquées ou arrondies, appartenant à D_1, D_2 et D_3 et présentées dans le tableau 2 ci-après.

D_i	D_1		D_2	D_3	Total
Résultat donné du calcul 1	23, 8 tronqué	23,9 arrondi	23,87 arrondi	23,872 tronqué	
Effectifs	1	4	7	1	13

Tableau 2. Les résultats donnés selon D_i dans l'institution française

Seuls les deux nombres 23,8 et 23,872 sont des valeurs tronquées. Les autres sont des valeurs arrondies.
On peut en conjecturer des règles du contrat didactique de calcul instrumenté par la calculatrice, différentes selon l'institution :

- en France, on est autorisé à donner comme résultat d'un calcul instrumenté par la calculatrice des valeurs arrondies ou tronquées à partir du nombre décimal affiché sur l'écran
- au Viêt-nam, le résultat d'un calcul instrumenté par la calculatrice est le nombre décimal tel qu'il apparaît sur l'écran de la calculatrice.

Nous allons maintenant approfondir notre analyse en analysant les procédures de calcul des élèves, procédures que nous considérons comme routinières : quelle est la place et quel est le des mémoires dans ces procédures ?

II.1.2. Quelles sont les procédures de calcul des élèves ?

Dans le chapitre C2, nous avons mis en évidence deux modes de fonctionnement de la mémoire Ans :

- elle peut fonctionner dans un calcul sans que la touche « Ans » soit pressée. Nous notons ce fonctionnement de la mémoire Ans « Ans~~Sto~~ » puisqu'il n'y a pas un stockage intentionnel d'un nombre dans cette mémoire pour le réutiliser dans le calcul ;
- elle peut fonctionner dans un calcul par l'appui intentionnel de la touche « Ans » et être réutilisée plusieurs fois pour l'exécution d'un même calcul. Nous notons ce fonctionnement de la mémoire Ans « Ans ».

Les fichiers historiques permettent de « rejouer » le calcul instrumenté de l'élève et d'avoir ainsi accès aux procédures des élèves. Nous avons pu récupérer au total 94 fichiers :

- 75 dans l'institution française sur un total de 86 élèves, soit 87,21% ;
- 19 dans l'institution vietnamienne sur un total de 60 élèves, soit 31,67%.

Le tableau 3 ci-après résume les analyses des fichiers historiques (94 élèves).

mémoires et stratégies / Classes	Sans mémoire Nd	Ans~~Sto~~ Nd	Ans Md_a	Variables Md_v	Total
2nd F1 et F2	47	19	0	0	66
1er F	7	0	0	2	9
Sous total	54 (72%)	19 (25,33%)	0 (0%)	2 (2,67%)	75
2nd V	7	2	0	1	10
1er V	6	3	0	0	9
Sous total	13 (68,42%)	5 (26,32%)	0 (0%)	1 (5,26%)	19
Total	67 (71,28%)	24 (25,53%)	0 (0%)	3 (3,19%)	94

Tableau 3. Effectifs selon les procédures de calcul et le fonctionnement des mémoires

Le tableau 3 montre les résultats suivants :

- *Sur les procédures de calcul*

L'algorithme de calcul direct est le seul utilisé comme nous l'avions prévu.

- *Sur le fonctionnement intentionnel des mémoires*

Seuls trois élèves (2 en France, 3 au Viêt-nam, soit 3,19%) utilisent une mémoire variable dans leur calcul. La mémoire Ans n'est utilisée par aucun élève.

Ces chiffres valident notre hypothèse de recherche : la calculatrice dans EMS fonctionne comme une machine arithmétique dans les pratiques routinières.

- *Sur le fonctionnement non intentionnel de la mémoire Ans*

25 élèves soit 26,6% utilisent la mémoire Ans~~Sto~~ pour exécuter un calcul.

Bien que le caractère représentatif de ces résultats puisse être discuté à cause du nombre des fichiers historiques récupérés dans chaque institution (87,21% en France contre 31,67% au Viêt-nam), d'autres résultats peuvent être établis concernant l'usage de la mémoire « Ans~~Sto~~ ».

- Les élèves français utilsent plus la mémoire « Ans~~Sto~~ » dans leur calcul que les élèves vietnamiens : 22,09% (19/86) contre 8,33% (8/60).

- Les 19 élèves français qui recourent à l'usage de la mémoire « Ans~~Sto~~ », sont tous des élèves de 2^{nde} : on peut avancer que cette pratique est un héritage du collège.

Rappelons que l'usage de mémoire « Ans~~Sto~~ » dans un calcul découpe un calcul en plusieurs sous calculs. Nous donnons trois exemples réalisés par les élèves de seconde. Pour le calcul 1, $2 \times x \times x + x + 1$, il y a 16 découpages possibles[1] en sous calculs.

Dans ces exemples, et ceux qui suivent, nous notons les touches appuyées par une écriture « arithmétique ».

+) 3,141 × 3,141 = × 2 = + 3,141 = +1 = ; (2^{nde} F1)
+) 2 × 3,141 × 3,141 = + 3,141 + 1 ; (2^{nde} F1)
+) 3,141 × 3,141 = Ans × 2 = + 3,141 = + 1 ; (2^{nde} F2)

Dans les deux premiers découpages, la touche « Ans » n'est pas pressée. Dans le troisième, elle est appuyée : nous l'avons cependant classé comme « Ans~~Sto~~ », cette touche n'étant pas réutilisée dans la suite de calcul.

En ce qui concerne les deux stratégies Md_a et Md_v nous constatons que :

- deux élèves (l'un en 1^{er} F, l'autre en 2^{nd} F) utilisent d'emblée les mémoires, attestant ainsi de leur instrumentation :

+) 3,141 Sto A 2 × A × A + A + 1 = ; (2^{nd} F) la mémoire A est considérée comme variable mathématique
+) 3,141 Sto A A × A = 2 × Ans = Ans + A = Ans + 1= ; (1^{er} F)

- le troisième élève (classe 1^{er} F) entre dans un véritable travail d'exploration et d'expérimentation sur les touches mémoires variables (entre crochets): après plusieurs essais et après stockage du nombre 3,141 dans la mémoire A, il trouve le résultat par un calcul direct. Puis, fort de ce résultat, il cherche à nouveau à calculer en faisant jouer un rôle « algébrique » à la mémoire A, ceci à deux reprises (en italique).

[A Sto 3.141 C/OFF C/OFF C/OFF AC/ON AC/ON A STO 3 3.141 A AC/ON 3.1 C/OFF AC/ON A STO 3 STO RCL A AC/ON AC/ON 3.. AC/ON 3.141 × 141 C/OFF C/OFF C/OFF 3.141 × 2 + 3.141 × C/OFF + 1 AC/ON A STO 3 A C/OFF C/OFF A STO 3 3.141 Ans C/OFF C/OFF C/OFF AC/ON C STO STO STO 3 STO STO STO A A AC/ON AC/ON AC/ON SHIFT SHIFT RCL RCL 333 ANS AC/ON AC/ON 3.141 STO STO STO A ANS ANS AC/ON A × A ANS ANS AC/ON 2 × 3.141 × 3.141 × C/OFF + 3.141 + 1 = AC/ON] *A × A = + A + 1 = AC/ON A × A × 2 + A + 1 = AC/ON* ;

II.1.3. Conclusion de l'analyse des pratiques routinières du calcul instrumenté

L'analyse *a posteriori* montre que ce calcul n'est pas problématique pour des élèves à ce niveau. Elle valide notre hypothèse formulée dans l'analyse *a priori* en ce qui concerne le fonctionnement de la calculatrice comme une machine arithmétique dans les pratiques routinières.

Dans les classes de seconde françaises et dans une moindre mesure dans les classes vietnamiennes, cette analyse révèle une pratique de calcul, utilisant la mémoire Ans~~Sto~~ : l'usage de cette mémoire découpe un calcul en sous calculs. Cette pratique, héritée du collège,

[1] On peut décrire l'expression $2x^2 + x + 1$ sous la forme (2 ; ×x ; ×x ; +x ; +1) ou (x ; ×x ; ×2 ; +x ; +1). Chaque découpage étant schématisé par une chaîne (2, ×x_n_1, ×x_n_2, +x_n_3, +1) ou (x, ×x_n_1, ×2_n_2, +x_n_3, +1) où n_i égale à 1 ou 0. Si $n_i = 1$, on découpe le calcul par le signe « = », si $n_i = 0$, on continue le calcul sans rien faire. Par exemple, une chaîne (2, ×x_1, ×x_0, +x_1, +1) donne le découpage 2 ×x = ; ×x +x = ; + 1. Nous avons donc $2^3 + 2^3$ chaînes, c'est-à-dire 16 découpages possibles.

a un caractère « privé » car n'étant nulle part enseignée dans EMS. La reconstitution des procédures permises par les fichiers historiques nous les a rendues visibles.

Dans les classes françaises, nous avons mis en évidence une règle du contrat didactique du calcul instrumenté par la calculatrice : un élève a le droit, dans un tel calcul, de donner un résultat approché, par arrondi ou par troncature, à partir d'un nombre affiché à l'écran.

II.2. Calcul 2

Nous rappelons le calcul 2 : $8x^3 + 6x^2 - 3x - 1$ avec x = 3,141759682.
Le résultat affiché est donc 296,888424, le résultat avec des chiffres de réserve dans la calculatrice est 296,88842400055[1], le résultat exact est 296,88842400055646923666050054 4.

L'enjeu de ce calcul est d'appuyer sur le moins de touches possibles : on demande aux élèves de noter le résultat et le nombre d'appuis de touches pour effectuer le calcul avec la calculatrice.

II.2.1. Quel résultat du calcul 2 est donné dans les productions écrites ?

Nous donnons dans le tableau 3 les effectifs selon que le résultat du calcul 3 est correct (y compris les valeurs approchées) ou incorrect.

Résultat / Classes	Correct		Incorrect	Pas de résultat	Total
	affichage	approché			
2nd F1 et F2	20	10	30	6	66
1er F	16	2	2	0	20
2nd V	25	1	11	5	42
1er V	15	0	3	0	18
Total	76	13	46	11	146

Tableau 3. Répartition des élèves selon l'exactitude de la réponse au calcul 2

Ce calcul est plus problématique que le précédent pour les élèves des deux institutions :

- 11 élèves (sur 146 élèves, soit 7,53%,) n'ont pas donné de résultat sur la fiche,
- 46 élèves (soit 31,51%) ont donné une réponse incorrecte.

Au total 39% des élèves des deux institutions ont rencontré des difficultés à effectuer le calcul 2 à l'aide de leur calculatrice ou d'Alpro.

Le saut informationnel organisé entre les calculs 1 et 2 a bien changé la complexité du calcul instrumenté.

Dans les deux institutions, ce calcul est plus problématique pour les élèves de 2nde que pour ceux de 1ère : 86,84% contre 51,85%. :

Cette analyse valide le contrat dans l'institution française dégagé de l'analyse précédente concernant les valeurs approchées ou tronquées des résultats apparus à l'écran de la calculatrice. Les mêmes 13 élèves de l'institution française prennent comme résultat des valeurs approchées ou tronquées contre un seul élève vietnamien en classe 2nd V.

Le seul observable concernant l'usage des mémoires, à partir des fiches élèves, est le nombre d'appuis de touches.

[1] Dans une telle calculatrice (TI 92, par exemple), la mémoire contient 14 chiffres au maximum pour un nombre. Dans d'autres, comme Casio 570MS, la mémoire contient 12 chiffres au maximum. Ce nombre devient 296,888424000.

L'analyse *a priori* a calculé qu'un nombre d'appuis de touches :

- compris entre 30 et 40 est attaché à une stratégie de calcul qui mobilisent les mémoires variables ou Ans ;
- supérieur à 80 est attaché à une stratégie de calcul qui n'utilisent pas les mémoires.

Dans la réalité, il nous faut tenir compte de la façon dont l'élève décompte ces appuis. Le tableau 4 ci-après donne une répartition des élèves selon le nombre n d'appuis de touches écrit sur leur fiche.

n ∈ / Classes	[20-30)	[30-40)	[40-50)	[50-60)	[60-70)	[70-80)	[80-90)	≥ 90	Total résultat	Sans réponse	Total
2nd F1 et F2	5	0	6	0	5	20	18	0	58	8	66
1er F	1	8	2	0	0	7	2	0	20	0	20
Sous total	6	12	8	0	5	27	20	0	78	8	86
2nd V	2	23	2	0	0	1	0	1	29	13	42
1er V	0	0	2	2	0	1	12	1	18	0	18
Sous total	2	23	4	2	0	2	12	2	47	13	60
Total	8	35	12	2	5	29	32	2	125	21	146

Tableau 4. Répartition des élèves selon le nombre n d'appuis de touches

L'examen des fichiers historiques disponibles montre que la donnée d'une valeur n du nombre d'appuis de touches compris entre 40 et 70 peut provenir aussi bien de procédures instrumentées sans mémoire qu'avec mémoire : nous ne retiendrons pour la suite que les intervalles extrêmes [20 ; 40) et n ≥ 70.

Le tableau 4 met en évidence les résultats suivants :

- Tout d'abord 13 élèves de 2nd V (sur 42, soit près de 31%) et 7 élèves de 2nd F2 (sur 33, soit 21%) ne donnent pas de nombre n (alors qu'ils ne sont que 5 en 2nd V et 4 en 2nd F2 à ne pas donner le résultat). Deux raisons peuvent être envisagées : le temps d'exécution résultant de la complexité du calcul ou les difficultés de manipulation de la calculatrice.
- Si nous considérons qu'un nombre n d'appuis de touches, supérieur à 70, comme un indice très probable d'une *stratégie sans mémoires,* au moins 63 élèves (sur 125 ayant donné un résultat, soit 50%) n'ont utilisé aucune mémoire dans leur calcul, malgré l'enjeu didactique de minimiser le nombre de touches et le coût du calcul. Cet effectif important montre la résistance de la calculatrice machine arithmétique. Cette résistance semble plus importante dans l'institution française (47 élèves sur 78 ayant répondu soit 54,6%) que dans l'institution vietnamienne (16 élèves sur 60, soit 26,6 %)
- Si nous considérons qu'un nombre n d'appuis de touches compris entre 20 et 40 est l'indice *assez probable* d'une *stratégie avec mémoires*, 43 élèves des deux institutions sur 125 ayant donné une réponse (soit 34%) *au plus* ont utilisé les mémoires dans leur calcul contre 3,19% dans le calcul précédent. *Assez probable*, car certains élèves peuvent oublier de décompter les 11 appuis pour l'entrée du nombre 3,141759682.
  Dans l'institution vietnamienne, il y a plus d'élèves ayant utilisé des mémoires de leur calculatrice que dans l'institution française : 41,67% contre 20,93%.
- Dans l'institution française, il y a plus d'élèves de 1er qui mobilisent des mémoires dans leur calcul qu'en 2nd : 45% contre 13,64%. Les tâches de calcul présentes au Lycée semblent faire évoluer, en France, l'instrumentation de la calculatrice par les élèves.
  L'évolution semble aller dans le sens contraire dans l'institution vietnamienne : 59,52% en 2nd (c'est-à-dire sortant du collège) contre 0% en 1er. L'absence de tâches de calcul instrumenté, repérée dans le chapitre C2, semble effacer les quelques connaissances instrumentales ayant pu être acquises au collège.

Le tableau 5 ci-après présente les effectifs des résultats incorrects au calcul 2 selon que le calcul a été effectué sans ou avec mémoires. L'absence d'usage des mémoires est déterminée soit à partir des fichiers historiques soit, pour les élèves qui ne travaillent pas sur Alpro, à partir de la taille du nombre n : $n \geq 70$ (tableau 4). Toutes les procédures des élèves de 2nd F observées ont pu être identifiées grâce à leur fichier historique. Le seul élève (sur 13) de la classe de 2nd V qui n'a pas donné de résultat travaille sur Alpro. L'examen de son fichier montre qu'il n'a pas utilisé les mémoires pour son calcul instrumenté.

Mémoire / Classes	Résultat incorrect				Résultat donné
	~~M~~	M	Sans réponse	Total	~~M~~
2nd F1 et F2	27	3	0	30	38
1er F	1	1	0	2	9
Sous total	28	4	0	32	47
2nd V	6	2	3	11	2
1er V	2	1	0	3	14
Sous total	8	3	3	14	16
Total	36	8	3	46	63

Tableau 5. Répartition des résultats incorrects au calcul 2 selon l'usage de mémoire

En classe de 2nd F, 27 élèves (sur 38) n'ayant pas utilisé de mémoire dans le calcul 2 donnent un résultat incorrect. Dans les autres classes, l'effectif des résultats incorrects (1er F et 1er V) et l'effectif des calculs sans mémoire (classe 2nd V) ne sont pas pas assez grands pour pouvoir tirer une information significative.
Le blocage des touches « x^2 » et « x^ » oblige à la récriture de l'expression donnée en $8 \times x \times x \times x + 6 \times x \times x - 3 \times x - 1$. Ce travail de réécriture donne lieu à peu d'erreurs. Seuls quatre élèves de 2nd F n'ont pas réussi cette réécriture : ils ont transformé à l'écran l'expression algébrique $8v^3 + 6v^2 - 3v - 1$ en $8 \times v \times 3 + 6 \times v \times 2 - 3 \times v - 1$.

Mais quel algorithme de calcul a servi à donner le résultat ? Quel type de mémoires a été utilisé, en particulier pour les élèves qui fournissent un nombre d'appuis de touches compris entre 40 et 70 ?
Nous allons analyser les fichiers historiques récupérés.

II.2.2. Quelles sont les procédures de calcul des élèves ?

L'analyse des fichiers historiques du calcul 3 montre l'absence attendue des algorithmes de calcul Ap et Ah (cf. chapitre C2). De même, aucune transformation algébrique de l'expression n'est observée en accord avec le contrat du calcul instrumenté avec la calculatrice dans lequel nous avons volontairement placé les élèves.
Les fichiers historiques récupérés permettent de reconstituer la succession des touches effectivement appuyées pour un calcul, que nous appellerons « programme en acte ».

Le tableau 5 ci-après met en évidence les effectifs selon les stratégies et les institutions, calculés à partir de l'analyse des fichiers récupérés (94 sur 146) :

mémoires et stratégies / Classes	Sans mémoire Nd	Ans~~Sto~~ Nd	Ans Md_a	Variables Md_v	Total
2nd F1 et F2	34	20	3	9	66
1er F	4	0	0	5	9
Sous total	38 (50,66%)	20 (26,67%)	3 (4%)	14 (18,67%)	75
2nd V	4	2	3	1	10
1er V	4	1	4	0	9
Sous total	8 (42,10%)	3 (15,79%)	7 (36,84%)	1 (5,26%)	19
Total	46 (48,94%)	23 (24,47%)	10 (10,64%)	15 (15,96%)	94

Tableau 6. Répartition des effectifs selon le type de stratégies et les mémoires utilisées

Malgré l'enjeu didactique de minimiser le nombre de touches et le coût du calcul, la moitié des élèves ayant travaillé sur Alpro (48,94%) utilise la calculatrice comme une machine arithmétique (sans mémoire effaçable).
Seuls 26,60% d'élèves ayant travaillé sur Alpro ont utilisé des mémoires. Cet effectif est plus faible que celui obtenu à partir de l'observable nombre d'appuis n des touches (tableau 4 : 34,4%), écart provenant des erreurs de décompte du nombre n d'appuis (oubli de compter les 11 appuis de l'entrée du nombre 3,141759682). Ces deux résultats confirment la résistance du fonctionnement de la calculatrice comme machine arithmétique.

Récapitulons les différentes procédures des élèves reconstruites à partir des fichiers historiques.

Les stratégies Nd – sans mémoire machine (mémoire papier)

Sur 46 d'élèves n'ayant utilisé aucune mémoire « machine » (c'est-à-dire Ans~~Sto~~, Ans, A, B, C) :

- 43 tapent leur calcul sur leur calculatrice en une seule fois donc avec au minimum 80 touches.
- 3 élèves (soit 6,52%) utilisent la mémoire « papier » pour noter les résultats intermédiaires. Nous donnons ci-après les procédures de ces trois élèves.

Une élève (2nd F2) a procédé comme suit :

1er essai : taper 3,141759682 × 3,141759682 × 3,141759682 × 8 = [ce qui donne 248,0897796]; noter le nombre 248 sur le papier ; taper 3,141759682 × 3,141759682 × 6 [59,22392339]; noter le nombre 59,2 sur le papier ; taper 3 × 3,141759682 = [9,425279046] ; noter le nombre 9,4 sur le papier ; taper 248 + 59,2 – 9,4 – 1 = [296,8] ;
2e essai : taper 3,141759682 × 3,1417596~~6~~82 × 3,14~~1~~1759~~6~~82 + 6 × 3,1417~~5~~9682 × 3,14~~1~~175~~9~~682 – 3 × 3,141759682 – 1 = [*297,46020952338*; ce qui est barré est un chiffre du nombre v que cette élève n'a pas tapé] ;
3e essai : taper 8×3.141759682×3.141759682×3.141759682+6×3.141759682×3.141759682 − 3×3.141759682−1 = [296.888424] ;

Le résultat écrit sur la fiche est « 297,5 ». Cette élève fait alterner dans son calcul des phases de frappe sur Alpro et de mémorisation des RI arrondis sur le papier. Les deux autres essais se font sans noter les RI. Elle choisit de retenir le résultat du second essai (en italique) pour lequel elle fait 6 fautes de frappe sans s'en rendre compte. Dans sa logique de d'approximation (contrat didactique du calcul instrumenté), elle donne comme résultat du calcul 2 un nombre arrondi (le résultat du calcul 1 était 23,9).

Un second élève (2nd V) procède comme suit :

Taper 3 × 3,141759682 = [ce qui donne 9,425279046] ; noter le nombre 9,425279046 sur le papier ; taper 3,141759682 × 3,141~~7~~59682 = [9,870142163] × 6 = [59,22085298] ; noter le nombre 59,22085298 ; […] taper 3,141759682 × 3,141759682 × 3,141759682 × 8 = [248,0897796] + *59,22085298* = [307,3106326] – *9,425279046* – 1 = [296,8853536] ;

Cet élève note deux RI (en italique) sur le papier et fait une erreur de recopie pour le résultat du calcul 2 : « 296,88424 » au lieu du nombre affiché (296,8853536).
Cet élève utilise une fois la mémoire Ans~~Sto~~ dans son calcul. Il fait aussi une faute de frappe sans s'en apercevoir, ce qui explique son résultat incorrect.

La troisième élève (2^{nd} F1) effectue 2 essais avant d'arriver au résultat du calcul.
Le premier avec une faute de frappe dans l'entrée du nombre v : « 248,0897796 + 59,2239234 + 9,425279046 − 1= » [315,738982] ;
Le deuxième essai : « 248,0897796 + 59,2239234 − 9,425279046 − 1 = » [296.888424].

Ces exemples montrent la place des fautes de frappes ou de recopie comme sources d'erreurs pour le résultat final. Le nombre v (3,141759682) comporte 9 chiffres différents, rendant difficile non seulement une « mémorisation mentale », mais aussi son entrée en machine ou sa recopie. Le fait que 73,68% des résultats incorrects pour le calcul 2 proviennent de calcul sans mémoire peut s'expliquer ainsi.

Le stockage du nombre dans une mémoire diminue ces risques d'erreurs et donc augmente la fiabilité du calcul.

La stratégie Nd - Ans~~Sto~~

23 élèves sur 94 (24, 47%) utilise la mémoire Ans~~Sto~~ en atomisant leur calcul en sous calculs. Sur ces 23 élèves, 3 seulement sont vietnamiens.
Cet effectif confirme que cette pratique « privée » de calcul est propre aux élèves de l'institution française.
Comme pour le calcul 1, elle est davantage présente en 2^{nd} qu'en 1^{er}, ce qui confirme qu'elle a sans doute était apprise dans des tâches de calcul du collège. Elle supplante dans l'institution française la stratégie Md_a,(3 élèves contre 20), alors que c'est plutôt l'inverse au Viêt-nam (7 contre 3).

Voici quelques exemples que nous avons reconstitués à partir des fichiers historiques :

+) 3,141759682 × 3,141759682 × 3,141759682 = × 8 = + 3,141759682 × 3,141759682 × 6 = - 3 × 3,141759682 = - 1 ;
+) 8 × 3,141759682 × 3,141759682 × 3,141759682 = 3,141759682 × 3,141759682 = × 6 = - 3 × 3,141759682 = - 1 ;

Cet usage de la touche mémoire « Ans » peut induire des erreurs résultant du découpage :

+) 3,141759682 × 3,141759682 × 3,141759682 = × 8 = + 6 = × 3,141759682 × 3,141759682 = - 3 = × 3,141759682 = - 1 ;

L'expression à calculer $8x^3 + 6x^2 – 3x – 1$ est transformée par le découpage en $((8x^3 + 6)x^2 – 3)x – 1$.

Les stratégies à mémoires

Nous avons voulu que l'usage de mémoire dans le calcul soit lié à la volonté de réduire le coût de calcul, en l'occurrence le nombre d'appuis de touches.
Nous allons examiner quel programme en acte, *c'est-à-dire quelle succession de touches effectivement appuyées*, les élèves adressent à leur machine à mémoire (A, B, C et Ans) qu'est pour eux Alpro.

Les exemples de programmes en actes non écrits montrent la présence d'une phase exploratoire ou expérimentale (entre crochets dans les exemples), représentant un processus d'instrumentation des touches mémoires.
Nous donnons ci-après deux exemples de phases exploratoires pour chacune des mémoires Ans et variables.

La stratégie Md_a : mémoire Ans

+) [3,141759682 × 3,141759682 × 3,141759682 = [31.01122246] × [Sur écran d'Alpro : *Ans*] 8 = [248.0897796] + 6 = [254.0897796] × 3,141759682 = [798,2890251] AC/ON AC/ON - 3Ans [Message envoyé à l'écran par Alpro : *Erreur Synt. Expression non Valide*] AC/ON 3,141759682 × 3,141759682 × 3,141759682 = [31.01122246] × 8 + 6 × 3,141759682 AC/ON AC/ON 3,141759682 = = = 3 × Ans = 9,425279046 3,141759682 = *Ans × Ans × Ans × 8 + 6 × Ans × Ans – 3 × Ans + 1* = ; (1er V)

La stratégie Md_v – mémoire variable

+) [8 × 3,141759682 A AC/ON A = AC/ON A 3,141759682 = 8 × 3,141759682 × 3,141759682 × 3,141759682 AC/ON A A AC/ON Sto A AC/ON A STO 3 Sto Sto Sto = 3 AC/ON] *3,141759682 Sto A AC/ON 8 × A × A × A + 6 × A × A – 3 × A + 1* = ; (2nd F2)

Ces deux élèves commencent tous les deux par un calcul direct :
+) le premier rencontre la mémoire Ans par appui sur la touche « = », puis tente de l'utiliser dans son calcul ;
+) le deuxième, en appuyant sur les touches mémoires variables, a accès aux informations données par Alpro sur ces touches : « **A.** Stocker une valeur dans une mémoire variable A » et cherche à stocker le nombre v dans une telle mémoire ; pour cela, il explore une autre touche « Sto ».

L'aboutissement à un programme en actes utilisant un stockage intentionnel dans une mémoire se fonde sur cette phase d'exploration.

Le tableau 7 montre quels sont les élèves pour lesquels existe ou n'existe pas une phase exploratoire.

Instrumentation / Stratégie / Classes	Sans phase exploratoire		Phase exploratoire		Total
	Md_a	Md_v	Md_a	Md_v	
2nd F1 et F2	2	6	1	3	12
1er F	0	4	0	1	5
Sous total	2	10	1	4	17
2nd V	2	1	1	0	4
1er V	0	0	4	0	4
Sous total	2	1	5	0	8
Total	4	11	6	4	25

Tableau 7. Répartition des procédures avec mémoire selon la phase exploratoire

L'absence de phase exploratoire permet d'affirmer que 15 élèves (sur 25) connaissent déjà les touches mémoires et leur usage, alors que 10 élèves les découvrent ou les redécouvrent, 4 pour les mémoires variables et 6 pour la mémoire Ans.

II.3. Conclusion de l'analyse *a posteriori* des calculs 1 et 2 la situation 1

Le seul algorithme présent dans les calculs instrumentés des élèves est l'algorithme de calcul direct Ad, ce que nous avions prévu dans notre analyse *a priori*.

Si le premier calcul n'est pas problématique pour la plupart des élèves observés, le deuxième calcul l'est, en revanche, beaucoup plus (46 erreurs et 11 sans réponse sur 145 élèves observés). Rappelons que nous l'avions voulu en organisant un saut informationnel entre les deux calculs : les valeurs du nombre v et du degré n ont été changées, mais aussi l'enjeu didactique du calcul : pour le deuxième calcul il ne s'agit pas seulement de donner le résultat mais aussi de minimiser le nombre d'appuis sur les touches de la calculatrice.

Les résultats des élèves aux calculs 1 et 2 valident notre hypothèse selon laquelle le rapport à la calculatrice dans EMS est essentiellement celui d'une machine arithmétique pour ce type de calcul, ce rapport résistant aux conditions du calcul 2.

Deux caractéristiques propres au calcul instrumenté *dans l'institution française* sont rendues visibles grâce aux fichiers historiques d'Alpro :

- Une règle du contrat didactique du calcul instrumenté : on a le droit d'écrire la valeur approchée, arrondie ou tronquée, du nombre affiché à l'écran de la calculatrice, comme résultat d'un calcul instrumenté.
- L'usage de la mémoire Ans~~Sto~~ dans un calcul : cet usage découpe un calcul en sous calculs. Cette pratique « privée », car n'existant pas dans EMS, est un héritage des pratiques de calcul instrumenté du collège et tend à disparaître en 1ère S. Ce découpage peut être source d'erreur comme nous l'avons montré.

Dans l'institution vietnamienne, la moitié des élèves sortant du collège (en classe 2nd) utilise la mémoire alors que très peu d'élèves en classe de 1er l'utilisent. Le vide didactique affectant les tâches de calculs instrumentés repéré dans le chapitre C2 semble avoir pour conséquence d'effacer les quelques connaissances instrumentales acquises au collège.

La plupart d'élèves de 2nd dans l'institution française n'ayant pas utilisé les mémoires dans le calcul 2 donnent un résultat incorrect. Les causes des erreurs sont essentiellement l'entrée du nombre dans la calculatrice et la recopie des résultats que le stockage d'un nombre dans une mémoire élimine.

Ils montrent qu'un processus d'instrumentation des touches mémoires se met en place pour les élèves qui les rencontrent pour la première fois (10 sur 25 élèves).

II.4. Analyse de la phase d'institutionnalisation, suite au calcul 2

À la fin des calculs 1 et 2, l'enseignant ramasse les fiches de réponses, puis note les résultats attendus des deux calculs au tableau sans commentaire.
Il demande ensuite à un(e) élève qui a :

- écrit la réponse correcte du calcul 2
- le plus petit nombre d'appuis de touches

de venir sur l'ordinateur « public » connecté à un vidéo-projecteur pour exécuter publiquement son calcul instrumenté.
Durant cette exécution publique, un autre élève (ou toute la classe) décompte en même temps le nombre des touches appuyées.

Voici les procédures de calcul des élèves désignés par l'enseignant dans chacune des 7 classes

Classes		Programme de calcul en actes
2nd F1	Groupe 1	3,141759682 Sto A 8 × A × A × A + 6 × A × A – 3 × A – 1 = ;
	Groupe 2	3,141759682 = 8 × Ans × Ans × Ans = + 6 × 3,141759682 × 3,141759682 = - 3 × 3,141759682 = ;
2nd F2	Groupe 3	3,141759682 = Ans × Ans × Ans = × 8 + 6 × 3,141759682 × 3,141759682 – - 3 × 3,141759682 – 1 ;
	Groupe 4	3,141759682 Sto A 8 × A × A × A + 6 × A × A – 3 × A – 1 = ;
1er F		3,14759682 Sto A × A × A × A × 8 + 6 × A × A – 3 × A – 1 = ;
2nd V		3,141759682 = 8 × Ans × Ans × Ans + 6 × Ans × Ans - 3 × Ans - 1= ;
1er V		3,141759682 = 8 × Ans × Ans × Ans + 6 × Ans × Ans - 3 × Ans - 1 = ;

Tableau 8. Les programmes en actes rendus publique selon les classes

Nous avions prévu que l'enseignant se contente de commenter l'usage des mémoires en expliquant l'économie qui en résulte du point de vue du nombre d'appuis de touches, en disant par exemple : « vous voyez, pour un même calcul, on peut passer de 80 à 31, 31 touches » (2nd F2 et 1er V).

L'enseignant de la classe de 1ère F a pris l'initiative de présenter les deux fonctionnements possibles de Ans~~Sto~~ et Ans comme mémoire du dernier résultat réutilisable. Par contre, comme les autres enseignants, il ne fait que mentionner la mémoire A alors qu'elle est utilisée dans le programme de calcul rendu public dans sa clase.

> Enseignant : Les résultats précédents c'est-à-dire lorsqu'on appuie sur la touche « = », on a enregistré le résultat de la calculatrice dans une mémoire qu'on appelle la mémoire Ans et on peut le rappeler en appuyant sur la touche « Ans ». Mais on n'a même pas besoin de la rappeler. Il suffit comme le suivant, de faire une opération directe. Par exemple, je vous donne, je veux faire 3 fois 4 plus 2, comme ça (il écrit au tableau). Je peux faire directement mais je peux faire aussi comme ça. Je fais 3 fois 4 et égal. Le résultat est alors mis automatiquement dans une mémoire qu'on appelle Ans. Je peux faire ensuite « Ans » car je sais que la calculatrice retient ce nombre, 12. Ans comme « Answer » en Anglais, et « Réponse » en français. Mais je peux aussi faire directement + maintenant (il écrit + 2). Et dans ce cas-là, la calculatrice va afficher « Ans » (il ajoute Ans à gauche de « + 2 »), elle-même et je mets plus 2 égale 14

Au tableau l'enseignant a écrit :

> 3 × 4 + 2 = ; 3 × 4 = ; Ans + 2 = 14

> Enseignant : Et donc je sais qu'il y a la mémoire Ans qui travaille pas de la même manière, si j'ose dire que la mémoire A. Vous pouvez rappeler ce qu'est la mémoire A ?
> Élève : A c'est la valeur qu'on a « storé » dans la mémoire A…
> Enseignant : D'accord. Pour ceux qui n'ont pas encore utilisé la mémoire A ou ceux qui le connaissent peu. On vous propose de continuer le 2e exercice. Et je vous donne tout de suite la fiche de l'exercice.

Dans quelle mesure ce bilan sur les mémoires et l'économie qu'elles procurent ont-ils des conséquences sur les pratiques instrumentées des élèves dans la situation suivante du calcul 3 ?

III. Situation 1 – calcul 3

Pour ce calcul, les 146 élèves sont répartis en 72 binômes.
La situation dans laquelle se place le calcul 3 s'organise selon les trois phases classiques d'une situation de communication :

- une phase d'écriture du message à un autre binôme : un programme écrit en langage des touches d'Alpro ;
- une phase de décodage du message : exécution du programme écrit par un autre binôme sur Alpro - validation pragmatique d'un programme et réception du résultat de l'exécution de son programme ;
- une phase de rectification du programme suite à la phase de décodage du message.

III.1. Écriture d'un programme en langage des touches d'Alpro

Phase 1 ***(10 minutes)*** *: Vous disposez d'un stylo et de la calculatrice Alpro*
Sur la feuille navette (phase 1), écrivez un message le plus court possible à un autre binôme qui ne possède ni papier ni stylo. Il dispose seulement de la même calculatrice que vous. Dans le message n'indiquez que les touches de la calculatrice. En suivant vos instructions, ce binôme doit obtenir la valeur numérique de z à l'écran de la calculatrice.

Variante 1 : $z = y^2 - 7y + 9$ où $y = \frac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$ avec x = 1,257

Variante 2 : $z = y^2 - 5y + 7$ où $y = \frac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$ avec x = 1,254

III.1.1. Quels programmes de calcul sont écrits pour le calcul 3 ?

Programme majoritairement écrit à une machine à mémoires
Les calculs 1 et 2, la phase de bilan qui a suivi le calcul 2 combinés au saut informationnel provoqué par le calcul 3 (fonctions composées) ont, de façon prévisible, modifié le rapport à la machine calculatrice comme le montre le tableau 9 ci-après :

Mémoires / Classes	oui	non	Pas de programme	Total
2nd F1 et F2	24	8	0	32
1er F	10	0	0	10
Sous total	34	8	0	42
2nd V	19	1	1	21
1er V	9	0	0	9
Sous total	28	1	1	30
Total	62	9	1	72

Tableau 9. Répartition de binômes suivant les programmes écrits pour le calcul 3 (phase 1)

Contrairement au calcul 2, où l'usage intentionnel de mémoires n'est le fait que d'un quart environ des élèves observés, pour le calcul 3, 62 binômes (sur 72) écrivent des programmes de calcul avec stockage intentionnel dans des mémoires.
La proportion des binômes utilisant des mémoires dans l'institution vietnamienne (28 binômes sur 30) est légèrement plus importante que dans l'institution française (34 sur 42).

Quels programmes sont écrits à la machine arithmétique, c'est-à-dire sans mémoire ?
Les 10 programmes sans mémoire-machine sont de deux types :

- non respect de la consigne : formule mathématique (3 programmes) ou description dans le langage courant des actions (2 programmes) ;
- programme en langage des touches d'Alpro, sans touche mémoire mais avec mémoire papier pour les RI (5 programmes).

Nous donnons un exemple de chacun de ces programmes dans le tableau 10.

Pas de programme : formule mathématique (3 programmes)	Programme dans le langage courant (2 programmes)	Programme langage des touches Alpro et mémoire papier (5 programmes)
Résoudre $y = \frac{2 \times 1{,}254 \times 1{,}254 \times 1{,}254 + 15 \times 1{,}254 - 5}{32{,}1 \times 1{,}254 - 27{,}7534}$ $z = y \times y - 5 \times y + 7$ ↘ résultat de y (2nd F1)	Pour trouver z, il suffit de taper sur 2 le multiplier par x^3 sachant que x = 1,254 puis additionner 15 x est soustraire à 5 le tout diviser par 32,1 x soustraire 27,7534 (2nd F1)	$2 \times 1{,}254 \times 1{,}254 + 15 \times 1{,}254 - 5 \div 32{,}1 \times 1{,}254 - 27{,}7534 =$ $5{,}194856975 \times 5{,}194856975 - 5 \times 5{,}194856975 + 7 =$ (2nd F1)

Tableau 10. Exemples de programmes écrits à la machine arithmétique pour le calcul 3 (phase 1)

Quels programmes sont écrits à la machine avec mémoire ?
Nous avons prévu dans l'analyse *a priori* trois stratégies possibles pour le calcul $f \circ g(v)$ selon l'usage des mémoires : M_{av}, M_{vm} et M_v.
Le tableau 11 présente la répartition des 62 programmes écrits utilisant les touches mémoires, observés, selon la stratégie à laquelle ils se rattachent.

Programmes / Classes	M_a	M_{av}	M_{vm}	M_v	Total
2nd F1 et F2	10	3	8	3	24
1er F	0	7	2	1	10
Sous total	10	10	10	4	34
2nd V	9	7	1	2	19
1er F	7	2	0	0	9
Sous total	16	9	1	2	28
Total	26	19	11	6	62

Tableau 11. Programmes écrits à Alpro, machine à mémoires, dans la phase de codage

• *Linéarisation du calcul pour l'écriture du programme en langage des touches d'Alpro*

Écrire un programme oblige à linéariser la fraction rationnelle $g(x) = \frac{N(x)}{M(x)}$.
Cette linéarisation nécessite elle-même l'usage des parenthèses : $(2x^3 + 15x - 5) \div (32{,}1x - 27{,}7534)$. Or les touches parenthèses sont indisponibles. Comment les élèves font-ils face à cette difficulté de l'écriture du programme ?

- L'une des réponses est d'utiliser des mémoires séparées pour les nombres N(v) et M(v), puis d'effectuer la division ;
- Une autre réponse est la mémoire papier, comme nous l'avons déjà vu précédemment.

Par exemple :

1,257 Sto A

4 × A × A – 9 × A + 21 = 17,63148637 Sto B
31,2 × A – 26,7184
7184 = 12,5
B ÷ 12.5 = 1,41051891 Sto C
C × C – 7 × C + 9 = 1,115931226

- Une autre réponse (12 binômes sur les 32 de 2nd F) est d'oublier la nécessité d'introduire des parenthèses. Ci-après un exemple d'une telle erreur :

1,254 Sto A
2 × A × A × A + 15 × A – 5 ÷ 32,1 × A – 27,7534 Sto B
B × B – 5 × B + 7

Le tableau 12 ci-après donne les effectifs des réponses mémoire papier ou oubli des parenthèses:

Classes	2nd F1 et F2	1er F	2nd V	1er V	Total
Oubli des parenthèses	**12**	2	2	1	17
Mémoire papier	4	2	**12**	**6**	24

Tableau 12. Nombres de binômes à propos du blocage des touches parenthèses dans la phase 1

La majorité des binômes ayant oublié des parenthèses provient des classes de 2nd F. Le travail nécessité par la linéarisation du calcul montre la fragilité des pratiques algébriques pour des élèves sortant du collège dans l'institution française. Ce travail semble plus outillé algébriquement en classe de 1er dans l'institution française.

Dans l'institution vietnamienne où l'algèbre domine les pratiques de calculs au collège et au Lycée, peu de binômes oublient les parenthèses dans la réécriture. Par contre, plus de la moitié des binômes vietnamiens utilisent la mémoire papier pour répondre au problème des parenthèses.

• *L'usage exclusif de la mémoire Ans envers et contre tout*

La majorité des programmes (26 M_a et 19 M_{av}, soit 45 programmes sur 62) utilisent la touche Ans dans leur calcul. L'instrumentation de la touche mémoire Ans commencée lors du calcul 2 se fortifie dans le calcul 3 alors que l'instrumentation des touches mémoires variables A, B, C s'avère plus problématique.
26 programmes utilisent la touche Ans de façon exclusive alors que nous l'avions rendu impossible par le blocage des touches parenthèses et le choix de la nature de la fonction g dans la fonction composée f∘g.
Comment, pour l'écriture de ces 26 programmes, les binômes ont-ils réussi à utiliser seulement la mémoire Ans, malgré son impossibilité ? A quels palliatifs ont-ils eu recours ?
Deux types de palliatifs sont présents :

- mémoire papier (22 programmes)
- changement indiqué par des notations écrites dans le programme du contenu de la mémoire Ans (4 programmes)

Nous donnons dans le tableau 13 des exemples de ces deux palliatifs :

RI noter dans le message : mémoire papier (22 programmes)	Changement du contenu des mémoires Ans par des notes sur le programme (4 programmes)
1,257 = 4 × Ans × Ans × Ans – 9 × 1,257 + 21 = ÷ 12,5 = AC/ON Ans × Ans – 7 × Ans + 9 = (Classe 1er V)	1. Entrer x = 14,257 puis appuyer sur = : Ans(1) 2. Faire 4 × Ans(1) × Ans(1) × Ans(1) – 9 × Ans(1) + 21 = Ans(nouveau) (2) 3. Ans(nouveau)(2) : 31,2 × 1,257 – 26,7184 = y = Ans(3) 4. Ans(3) × Ans(3) – 7 × Ans(3) + 9 = z (Classe 2nd V)

Tableau 13. Deux types de palliatifs pour l'usage exclusif de la mémoire Ans (phase 1)

L'usage Ans~~Sto~~ mentionné dans l'analyse des calculs 1 et 2, disparaît au profit du stockage intentionnel d'un nombre dans Ans par l'appui sur la touche « = » et de la réutilisation dans la suite du calcul.

• *Les programmes de calculs M_{vm} et M_{av} : un nombre et un seul dans une mémoire*

Nous avons caractérisé dans l'analyse *a priori* trois variantes possibles pour la stratégie $M_{av.}$:

- Variante (A, B, C et Ans) : *stockage du nombre v dans la mémoire variable A*, calcul de M(v), stockage de M(v) dans B, calcul de N(v), *stockage du N(v) dans C*, calcul de B ÷ C, *stockage de B ÷ C dans la mémoire Ans*, calcul de z.
- Variante (A, B, Ans et C) : *stockage du nombre v dans la mémoire variable A*, calcul de M(v), *stockage de M(v) dans B*, calcul de N(v), *stockage du N(v) dans Ans*, calcul de Ans ÷ B, *stockage de B ÷ C dans la mémoire* C, calcul de z.
- Variante (A, B, Ans et Ans) : *stockage du nombre v dans la mémoire variable A*, calcul de M(v), *stockage de M(v) dans B*, calcul de N(v), *stockage du N(v) dans Ans*, calcul de Ans ÷ B, stockage de Ans ÷ C dans la mémoire Ans, calcul de z.

Dans les deux premières variantes, chaque mémoire contient seulement un nombre durant tout le calcul. On peut considérer qu'une mémoire variable dans ce cas est une variable mathématique.
La mémoire Ans dans la 3e variante n'a pas le même statut, puisqu'elle contient successivement deux nombres qui sont à chaque fois le dernier résultat d'un calcul. Cette variante mobilise le moins grand nombre de touches mémoires. C'est en cette dernière

variante que l'instrumentation de la touche mémoire Ans se montre la plus nette : elle consiste en effet à reconnaître l'occasion de l'utiliser ou de ne pas l'utiliser.

Variantes de M_{av} / classes	A,B,C,Ans	A,B,Ans,C	A,B,Ans,Ans	Total
2nd F1 et F2	0	1	0	1
1er F	1	4	0	5
Sous total	1	5	0	6
2nd V	2	1	1	4
1er V	0	0	1	1
Sous total	2	1	2	5
Total	3	6	2	11

Tableau 14. Répartition des programmes selon les trois variantes de M_{av} dans le calcul 3

Le tableau 14 montre la rareté de la variante (A, B, Ans, Ans) : 2 élèves sur 11.

• *Les programmes de calculs M_{vm} et M_{va} : un nombre et un seul dans une mémoire*

Dans les programmes de calcul M_{vm} et M_{va} (hors la variante (A, B, Ans, Ans)) chaque mémoire variable A, B, C ou Ans est utilisée comme une lettre désignant un nombre dans le calcul. Le contenu de ces mémoires reste inchangé durant l'ensemble du programme de calcul. Plus de la moitié des programmes (34 sur 62) de calcul ont ces caractéristiques.
Pour les binômes ayant écrit ces programmes, une mémoire sert à stocker un et seulement un nombre durant le processus de calcul.
Nous donnons ci-après trois exemples de tels programmes. Nous indiquons entre parenthèses les mémoires utilisées.

Programme (A, B, Ans, C)	Programme (A, B, C, B ÷C)	Programme (A, B, C)
1,254 Sto A 32,1 × A – 27,7534 Sto B 2 × A × A × A + 15 × A – 5 = Ans ÷ B = Ans Sto C C × C + 5 × C – 5 × C + 7 = (Classe 11er F)	1,254 Sto A 2 × A × A × A + 15 × A – 5 Sto B 32,1 × A – 27,7534 Sto C B ÷C × B ÷C – 5 × B ÷ C + 7 = (Classe 2nd F2)	1,254 Sto A 2 × A × A × A + 15 × A – 5 ÷ 32,1 × A – 27,7534 Sto B B × B – 5 × B + 7 = (Classe 2nd F1)

Tableau 15. Trois exemples de programmes écrits utilisant les mémoires comme variables mathématiques

• *Le langage des touches d'Alpro*

Seulement cinq messages (tableau 10) ne sont pas écrits dans le langage des touches d'Alpro, ou *langage d'Alpro*.
L'usage du langage d'Alpro dans les messages évite les implicites présents dans les messages sous forme de langage naturel et donc restreint les interprétations possibles du binôme récepteur. Le langage Alpro est un langage évolué qui continuera à évoluer en interaction avec les modifications de la machine dans les situations suivantes de l'ingénierie didactique.
Dans ce langage, la notation d'affectation, notée « Sto » dans les messages et « → » sur l'écran correspond à la nécessité du stockage intentionnel d'un nombre dans une mémoire pour le calcul 3. Rappelons que cette notation est fondamentale dans l'émergence de la notion variable informatique lors de l'écriture d'un programme en langage évolué (cf. le chapitre B3).

• *Emergence difficile de mémoires effaçables*

La stratégie Mv est un observable de l'effaçabilité des mémoires variables et donc de l'émergence de la notion de variable informatique.
Le nombre faible des binômes utilisant une même mémoire variable pour stocker intentionnellement des nombres différents (6 élèves sur 62 : tableau 11) montre que la

caractéristique d'effaçabilité des mémoires variables résiste aux conditions mises en place dans la situation 1 (nombre limité à trois des touches mémoires).
La mise en place d'une telle caractéristique sera l'enjeu didactique premier de la situation 2 de l'ingénierie didactique.

III.1.2. Conclusion de l'analyse de la phase 1, calcul 3

L'analyse des programmes écrits par les binômes lors de cette première phase montre d'une part la disponibilité des mémoires et d'autre part la présence des deux types de mémoires : mémoires variables et mémoire Ans dans les programmes de calculs des binômes.

Le fait que la mémoire Ans soit utilisée exclusivement ou conjointement avec des mémoires variables A, B, C dans la majorité des programmes faisant appel à des mémoires montre que le processus d'instrumentation de la touche Ans se poursuit et s'enrichit. Par contre, celle des touches mémoires variables s'avère plus problématique.
La notion de « mémoire effaçable » reste absente de la programmation de ce calcul. Pour la plupart des élèves ayant utilisé une mémoire, une mémoire ne peut stocker qu'un unique nombre durant tout le calcul. Les mémoires fonctionnent dans le programme écrit comme une variable mathématique.
Le blocage des parenthèses oblige à un travail de réécriture du calcul 2, en particulier à la linéarisation de la fraction rationnelle $g(x)=\frac{N(x)}{M(x)}$. Ce travail montre la fragilité des pratiques algébriques des élèves sortant du collège en France, plus de moitié des binômes de 2nd F oubliant le rôle des parenthèses

III.2. Rétroactions apportées par la phase de décodage

> *Phase 2* ***(5 minutes)*** *: Rangez votre stylo. Vous n'avez plus que la calculatrice Alpro*
> Exécutez, avec la calculatrice Alpro, toutes les instructions du message reçu, tant que c'est possible. Reprenez alors votre stylo pour recopier sur la feuille navette (phase 2) le dernier écran de la calculatrice.

Les rétroactions pour un binôme sont de deux ordres :

- En tant que récepteur d'un programme : les informations apportées par l'exécution du programme écrit par un autre binôme ;
- En tant qu'émetteur d'un programme : le message du dernier écran copié par le binôme récepteur après l'exécution de son propre programme ;

La majorité des programmes sont écrits sous forme d'une suite de touches d'Alpro. La plupart des programmes écrits ont donc pu être exécutés sur Alpro par le binôme récepteur.

Rappelons que seuls 5 messages de 2nd F (deux en langage courant et trois en formule mathématique) ne sont pas écrits en langage d'Alpro.
Comment les binômes récepteurs de 2nd F font-ils pour exécuter le programme reçu ?
Cette exécution aboutit à un programme en acte résultant de cette interprétation. Nous donnons, dans le tableau 16, les programmes reçus par 2 de ces 5 binômes (langage courant et formule mathématique) puis l'interprétation en termes de programme en actes, reconstitué à partir des fichiers historiques.

Programmes reçus et écran écrit après exécution du programme en actes par le binôme récepteur	Interprétation : Programme en acte du récepteur	Programmes écrits du récepteur (phase 1)
Pour trouver z, il suffit de taper sur 2 le multiplier par x^3 sachant que x = 1,254 puis additionner 15x et soustraire à 5 le tout diviser par 32,1x soustraire 27,7534 » - 27,0598376	1,254 Sto A AC/ON 2 × A × A × A + 15 × A – 5 = Sto B B ÷ 32,1 A = AC/ON B ÷ 32,1 × A = Ans – 27,7534	Taper 1,257 × Sto A AC/ON 4 × A × A × A – 9 × A + 21 Sto B AC/ON 31,2 – 26,7184 Sto C AC/ON B ÷ C Sto A AC/ON A × A – 7 × A + 9
$y = \frac{2x^3 + 15x - 5}{32,1x - 27,7534} = B$ A = 1,254 = x 1040,718665	1,254 Sto A 2 × A × A × A + 15 × A – 5 = ÷ 32,1 × A – 27,7534 = Sto B B × B – 5 × B + 7	1,257 Sto AC/ON A × A × A × 4 – 9 × A + 21 = Ans ÷ 31,2 – 26,7184 = Sto B B × B – 7 × B + 9

Tableau 16. Interprétation en termes de « programmes en actes » des messages non écrit en langage d'Alpro

Ce tableau montre que l'interprétation du programme reçu s'appuie sur l'instrumentation des touches mémoires qu'atteste le programme écrit par les récepteurs durant la phase 1

Quelles influences ont les rétroactions de l'exécution des programmes écrits sur la ré-écriture du programme ?

III.3. Rectification du programme suite à la phase de décodage

Phase 3 ***(10 minutes)*** *: Vous disposez d'un stylo et de la calculatrice Alpro*
Sur votre fiche navette qui vient de vous être rendue, proposez si nécessaire, un nouveau message.

Le résultat du travail de rectification

Le tableau 17 donne le résultat de la rectification, dans la phase 3, des programmes suite à la phase de décodage et relativement à l'usage des mémoires :

Mémoires / Classes	oui		non		Pas de réponse		Total
	Phase1	Phase3	*Phase1*	Phase3	*Phase1*	Phase3	
2nd F1 et F2	*24*	27	*8*	3	*0*	2	32
1er F	*10*	10	*0*	0	*0*	0	10
Sous total	*34*	37	*8*	3	*0*	2	42
2nd V	*19*	21	*1*	0	*1*	0	21
1er V	*9*	8	*0*	0	*0*	1	9
Sous total	*28*	29	*1*	0	*1*	1	30
Total	*62*	66	*9*	3	*1*	3	72

Tableau 17. Rectification des programmes de calculs suite à la phase de décodage

Quatre binômes rectifient leur programme écrit en utilisant les mémoires alors qu'ils ne l'avaient pas fait auparavant : ce sont 3 binômes de 2nd F, et 1 binôme de 2nd V. Leurs programmes écrits dans la phase 1 et phase 3 ainsi que le programme décodé dans la phase 2 sont donnés dans le tableau 18 ci-après :

Binômes	Phase 1 : programmes écrit	Phase 2 : programme décodé	Phase 3 : programme rectifié
Évolution **Nd → Ma** 1 binôme (2^{nd} V)	x = 1,254 $y = \frac{2x^3 + 15x - 5}{32,1x - 27,7534}$ $z = y^2 - 5y + 7$ z =	Entrer x = 1,257 → = → Ans Calculer 31,2 × Ans – 26,7184 → = → 12,5 Calculer 4 × Ans × Ans × Ans - 7 × Ans +9 → = → ? Prendre ? → ÷ → 12,5 → =	Entrer 1,254 = → Ans Calculer 2 × Ans × Ans × Ans + 15 × 1,254 – 5 = → ? Calculer 32,1 × Ans – 27,7534 = 12,5 y = ? ÷ 12,5 = ?? Calculer z = ?? × ?? – 5 × ?? + 7 =
2 binômes (2^{nd} F1)	4 × 1,257 × 1,257 × 1,257 - 9 × 1,257 + 21 ÷ 31,2 × 1,257 − 26,7184 = - 29,24085593 × 5× - 29,24085593 – 5 × - 29,24085593 + 7 =	1,254 = 2 × Ans × Ans × Ans + 15 × 1,254 - 5 ÷ 32,1 × 1,254 – 27,7534 = Ans × Ans – 5 × Ans + 7 =	2 × 1,257 × 1,257 × 1,257 + 15 × 1,257 – 5 ÷ 32,1 × 1,254 − 27,7534 = Ans × Ans - 5 × Ans – 5 × 1,94856975
Évolution **Nd→M_{vm}** 1 binôme (2^{nd} F1)	$y = \frac{2x^3 + 15x - 5}{32,1x - 27,7534} = B$ A = 1,254 = x	Taper 1.257 × Sto A AC/ON 4 × A × A × A – 9 × A + 21 = Ans ÷ 31,2 × A – 26,7184 = × Sto B B × B – 7 × B + 9 = z	1) 1,254 Sto A 2) 2 × A × A × A + 15 × A - 5 = 3) ÷ 32,1 × A – 27,7534 = 4) Sto B 5) B × B – 5 × B + 7 =

Tableau 18. Évolution des programmes dans les trois phases : vers l'usage des mémoires

Ce tableau permet de montrer comment l'information portée par l'exécution du programme écrit par l'autre binôme joue un rôle prépondérant dans la rectification du programme initialement écrit.
N'ayant pas utilisé de mémoire le binôme récepteur, à travers le décodage du programme à mémoire qu'il a reçu, entre dans un processus d'apprentissage des mémoires.
Le binôme récepteur ne fait pas qu'« imiter » l'usage des mémoires dans le programme décodé. Il peut écrire un programme « amélioré » à partir du programme reçu.
Par exemple, en utilisant la mémoire Ans comme dans le programme qu'il a décodé, le binôme de la classe de 2^{nd} V « distingue » différentes mémoires Ans par les symboles : « Ans », ? et ?? dans le programme rectifié. La place de ces symboles, après la touche « = », attestent que ce binôme a compris que le nombre contenu dans la mémoire Ans change dès que la touche « = » est pressée.

Le tableau 19 ci-après montre les rectifications des programmes par rapport aux stratégies M_a, M_{av}, M_{vm}, et M_v.

Programmes / Classes	M_a		M_{av}		M_{vm}		M_v		Total
	Phase1	Phase3	*Phase1*	Phase3	*Phase1*	Phase3	*Phase1*	Phase3	
2^{nd} F1 et F2	*10*	10	*3*	7	*8*	2	*3*	8	14
1^{er} F	*0*	0	*7*	8	*2*	1	*1*	1	10
Sous total	*10*	10	*10*	15	*10*	3	*4*	9	37
2^{nd} V	*9*	4	*7*	12	*1*	1	*2*	4	21
1^{er} V	*7*	5	*2*	3	*0*	0	*0*	0	8
Sous total	*16*	9	*9*	15	*1*	1	*2*	4	29
Total	*26*	19	*19*	30	*11*	4	*6*	13	66

Tableau 19. Rectification des programmes écrits à Alpro, machine à mémoires, suite aux rétroactions

Ce tableau ainsi que le tableau 17 permettent de formuler quelques constats :

• *L'instrumentation des touches mémoires A, B et C continue*

Le nombre de binômes écrivant un programme suivant la stratégie M_v fait plus que doubler, passant de 6 à 13. La validation pragmatique et la lecture des programmes à exécuter a conduit 7 binômes utilisant les mémoires A, B et C comme des variables mathématiques à les utiliser comme des mémoires effaçables, c'est-à-dire comme pouvant changer intentionnellement de contenus.

L'information apportée par l'exécution d'un autre programme que le sien est le facteur prépondérant permettant ces binômes de prolonger le processus d'instrumentation des touches mémoires A, B et C qu'ils avaient commencé dans la phase 1, vers le statut de mémoire effaçable comme le montre l'exemple (classe 2nd V) ci-après :

Phase 1 : programmes écrit	Phase 2 : programme reçu	Phase 3 : programme récrit
1,254 Sto A AC 2 × A × A × A + 15 × A – 5 = 17,75387013 32,1 × A – 27,7534 = 12,7194 Sto B 17,75387013 ÷ B = 1,395810347 Sto C Calcul de z C × C – 5 × C + 7 = 1,96923478	1,257 Sto A AC 4 × A × A × A - 9 × A + 21 Sto B AC 31,2 × A – 26,7184 Sto C AC B ÷ C Sto A AC A × A – 7 × A + 9 = 1,115931226	1,254 Sto A AC 2 × A × A × A + 15 × A – 5 = 17,75387013 Sto B 32,1 × A – 27,7534 = 12,5 Sto C B ÷ C = Sto C C × C – 5 × C + 7 =

Le programme dans la phase 1 du binôme est M_{vm} avec l'usage de la mémoire papier :il a évolué en M_v après avoir exécuté un programme M_v d'un autre binôme.

Cependant, M_v reste moins présent que M_{av} : 13 contre 30. Or, dans M_{av} (sauf l'invariante (A, B, Ans, Ans) les deux mémoires reçoivent le même statut dans le sens où elles ne contiennent qu'un seul nombre durant tout le calcul.

• *Évolution des mémoires variables A, B, C et Ans dans la procédure M_{av}*

Rappelons que dans les deux premières variantes (A, B, C et Ans) et (A, B, Ans et C), chaque mémoire contenant un seul nombre durant tout le calcul est conçu comme une variable mathématique.

Dans le tableau 20, nous donnons la répartition des programmes suivant les trois variantes de M_{av} dans le calcul 3. Les programmes utilisant des palliatifs sont classés dans la catégorie « autres ».

Variantes M_{av} / Classes	(A, B, C, Ans)		(A, B, Ans, C)		(A, B, Ans, Ans)		Autres	
	Phase1	Phase3	*Phase1*	Phase3	*Phase1*	Phase3	*Phase1*	Phase3
2nd F1 et F2	*0*	0	*1*	2	*0*	0	*2*	5
1er F	*1*	0	*4*	6	*0*	0	*2*	2
Sous total	*1*	0	*5*	8	*0*	0	*4*	7
2nd V	*2*	4	*1*	1	*1*	3	*3*	4
1er V	*0*	0	*0*	0	*1*	1	*1*	2
Sous total	*2*	4	*1*	1	*1*	4	*4*	6
Total	*3*	4	*6*	9	*2*	4	*8*	13

Tableau 20. Répartition des programmes selon les trois variantes de M_{av} et les palliatifs dans le calcul 3

Les trois variantes augmentent. Les programmes (A, B, Ans, C) passent de 6 à 9, les deux autres (A, B, C, Ans) et (A, B, Ans, Ans) augmentent à 4 programmes.

Les effectifs des programmes corrects utilisant les mémoires variables soit seules (M_v) soit conjointement à la mémoire Ans (M_{av}) est donné dans le tableau ci-après.

Stratégies / Classes	M_v		M_{av}	
	Phase1	Phase3	*Phase1*	Phase3
2nd F1 et F2	*3*	8	1	0
1er F	*1*	1	5	2
Sous total	*4*	9	6	6
2nd V	*2*	4	2	8
1er V	*0*	0	1	8
Sous total	*2*	4	3	1
Total	*6*	13	9	17

Tableau 21. Évolution des programmes M_v et M_{av} dans le calcul 3 de la phase 1 à la phase 3

Même si les rétroactions de la phase 2 (exécution du programme reçu et effet de l'exécution du programme écrit) ont permis une évolution de l'instrumentation des touches mémoires variables et Ans dans le calcul 3, il y a seulement 30 programmes corrects sur 72 à la fin de la phase 3.
Parmi les 30 programmes corrects, il y a tous les programmes M_v et seulement 17 programmes M_{av}, les erreurs provenant principalement de la linéarisation du calcul et de l'oubli des parenthèses comme nous allons l'étudier maintenant.

• Linéarisation du calcul pour l'écriture du programme en langage des touches d'Alpro : effet de l'indisponibilité des touches parenthèses

Rappelons qu'écrire un programme oblige à linéariser la fraction rationnelle $g(x)=\frac{N(x)}{M(x)}$, ce qui oblige algébriquement à recourir à des parenthèses qui sont indisponibles pour le programme. Le palliatif attendu est l'instrumentation des mémoires variables ou Ans.
L'examen des fiches montre que, pour certains binômes, cette instrumentation ne s'est pas faite et qu'ils recourent aux palliatifs déjà mis en évidence dans la phase 1 :

- Oubli des parenthèses;
- Mémoire papier pour l'un des résultats intermédiaires ;

Le tableau suivant montre l'évolution des binômes entre la phase 1 et la phase 3 :

Classes / Palliatifs	2nd F1 et F2		1er F		2nd V		1er V	
	Phase1	Phase3	*Phase1*	Phase3	*Phase1*	Phase3	*Phase1*	Phase3
oubli parenthèses	***12***	**13**	*2*	2	*2*	0	*1*	1
mémoire papier	*4*	4	*2*	1	***12***	**8**	***6***	**5**

Tableau 21. Évolution de binômes vis-à-vis des palliatifs aux parenthèses

Ce tableau montre peu d'évolution entre les phases 1 et 3 alors que la plupart de ces binômes utilisent des mémoires dans leur programme de calcul.
En effet, leur programme aboutit à un nombre et il n'y a pas dans la situation d'information qui invalide le résultat du calcul 3, en dehors d'une évaluation mentale de l'ordre ou du signe du résultat, ce qui n'est pas une pratique habituelle du calcul instrumenté.

III.4. Conclusion de l'analyse du calcul 3

Le saut opéré entre le calcul 2 et le calcul 3 a permis d'installer la grande majorité des élèves dans une problématique d'utilisation des mémoires variables et Ans de la calculatrice. Seuls trois élèves sont en échec.

Les rétroactions apportées par la phase de décodage, notamment l'exécution d'un programme écrit par l'autre binôme, sont l'un des facteurs essentiels qui permettent de prolonger le processus d'instrumentation des touches mémoires, déjà initié depuis la phase 1 ou pour certains, depuis les calculs 1 et 2.

L'un des effets les plus marquants du saut informationnel entre le calcul 2 et 3, est l'instrumentation de la mémoire Ans qui est utilisée avec un stockage intentionnel et la disparition conséquente du fonctionnement Ans~~Sto~~ pour ce calcul. L'autre effet est l'utilisation répétée de la mémoire Ans dans le calcul 3 qui atteste de l'instrumentation de la touche mémoire Ans dans la reconnaissance des occasions d'usage.

À la fin du calcul 3 presque tous les messages sont écrits en langage Alpro : suite de touches d'Alpro. L'affectation d'un nombre dans une mémoire variable, « Sto » en langage Alpro, est présente dans les deux tiers des programmes écrits. Or la notion d'affectation d'un langage évolué, en l'occurrence le langage Alpro, est un préalable à la mise en place de la signification de la notion de variable informatique (cf. chapitre B3).
Cependant, les mémoires variables A, B, C ainsi que la mémoire Ans sont considérées de façon majoritaire comme des variables mathématiques : elles ne peuvent contenir qu'un seul nombre durant tout le calcul. La caractéristique d'effaçabilité d'une mémoire est peu présente dans les programmes écrits même après rectification.

III.5. Phases de bilan du calcul 3

À la fin de la séance, l'enseignant demande à un binôme de venir présenter publiquement son programme de calcul sur la calculatrice Alpro publique (branchée sur un vidéoprojecteur).

Nous présentons les bilans :

- des deux classes de 1er F et 1er V, classes dans lesquels est expérimenté l'ensemble des situations de l'ingénierie didactique.
- Le bilan dans le groupe 4 des 2nd F
- Le bilan de 2nd F fait en classe entière à propos de l'usage des mémoires dans le calcul 3.

Classe de 1er F :

Le programme rendu public et le programme en acte reconstitué à partir du fichier historique est donné ci-après.

Programme écrit du binôme	Programme en actes (à partir du fichier historique)
1,254 Sto A 32,1 × A − 27,7354 Sto B 2 × A × A × A + 15 × A − 5 = Ans ÷ B = Ans Sto C C × C − 5 × C + 7 =	1,254 Sto A 32,1 × A – 27,7354 Sto B AC/ON 32,1 × A – 27,7534 Sto B 2 × A × A × A + 15 × A – 5 = Ans ÷ B = [1,42030961] AC/ON Ans Sto C AC/ON 2 × A × A × A ÷ AC/ON 2 × A A AC/ON 2 × A × A × A + 15 × A – 5 = [17,75387013] Ans ÷ B = [1,42030961 Ans Sto C C × C – 5 × C + 7 = [1,915731338]

Ce programme sur lequel s'est bâtit la phase de bilan est du type M_{av} ,variante (A, B, Ans, C).

Les interactions avec l'enseignant lors de la présentation publique

P[1] : Bon, je vais demander à Long. Voilà, de venir ici nous exposer le message. Allons-y. […] Vous voyez qu'il a stocké 1.254 dans la mémoire A. Alors, ça mérite de commenter. Qu'est ce qu'il est en train de faire ?
E1 : (en voyant que le nombre 27,7534 est entré au lieu de 27,5734) Non c'est pas ça
P : Ah, si en plus s'il ne fait pas ce qu'il a dit.
E1 : Non, c'est 7, 3, et 4.

[1] C'est Alain Birebent.

P : Alors. D'accord. Qu'est ce que vous avez fait ? Qu'est ce qui s'est passé ? Qu'est ce que vous pouvez faire ? Mais comme ça, vous allez effacer tout ce que vous avez fait. Ah, on recommence. Peut-être que non. Ecoutez moi bien. Ce qui est dans la mémoire A est toujours dans la mémoire A.
E2 : On recommence avec la mémoire B simplement.
Ppu : Très bien, très bien. La mémoire A n'a pas bougé. Il veut simplement changer le contenu de la mémoire A. Qu'est ce qui s'est passé ? Qu'est-ce que vous avez fait ?
E2: Par ce qu'ici il y a pas des parenthèses donc on…
P : Donc vous avez décidé de mettre le dénominateur de la fraction dans la mémoire B. D'accord … Donc là. Vous pouvez commenter ?
E2 : Donc là, c'est, c'est…on a calculé y.
P : Vous calculez y. Vous voulez dire plutôt calculer le numérateur et vous avez divisé le numérateur. D'accord ? Et vous avez utilisé la mémoire Ans pour le dénominateur.
E2: C'est Ans Sto C.
P : Mais est ce qu'il y a une erreur là. À cause de Ans ? Parce que le Ans a été changé.
E2 : Oui.
P : Il faut reprendre. Ah oui, il faut rester très vigilant des statuts des mémoires Ans, A, B, C. Ca c'est le dénominateur, d'accord ? Egale d'accord ?
E2: Divisé par…
P : Divisé par B…Et Ans Sto C. Donc le C c'est quoi là? Le C c'est y. D'accord ?
E1, E2 : 5 fois C plus 7.
P : Pour ceux qui calculent avec x = 1,257 c'est la valeur de z. Voilà, on va faire une toute petite pause s'il vous plaît […]

L'enseignant a publiquement justifié l'usage de la mémoire variable A comme réponse au problème posé par l'indisponibilité des parenthèses. Il se contente par la suite de mentionner la mémoire Ans dont il a déjà présenté deux usages dans la phase de bilan du calcul 2. Puis il accompagne le binôme sans donner d'autres commentaires sur la mémoire variable.
Notons que le binôme a utilisé correctement la mémoire Ans, d'abord pour stocker le nombre N(v) puis le nombre y(v). Il a dû recommencer suite à un doute déclenché par l'enseignant.

Classe 1er V

Nous présentons ci-après le programme écrit par le binôme et le programme en actes, reconstitué à partir du fichier historique.

Programme écrit du binôme	Programme en actes (à partir du fichier historique)
1,254 SHIFT STO A AC/ON A × A × A × 2 + 15 × A – 5 SHIFT STO B AC/ON 32,1 × A – 27,7534 = B : Ans = Ans × Ans – 5 × Ans + 7 = 1,915731330	AC/ON(5) 1,254 Sto A AC/ON AC/ON A × A × A × 2 + 15 × A × A C/OFF C/OFF − 5 = [17,75387013] Shift Sto B AC/ON 32,1×A−27,7534 = [12,5] B ÷ Ans = [1,42030961] Ans × Ans − 5 × Ans + 7 = [1,915731338]

Le programme écrit par de ce binôme et sur lequel s'est bâtie la phase de bilan est du type M_{av} variante (A, B, Ans, Ans). Ce binôme appuie toujours sur la touche « Shift » pour accéder à une touche mémoire variable alors que cela n'est pas nécessaire sur Alpro.

Les interactions avec l'enseignant lors de la présentation publique

P [1] : Je demande au groupe d'H et T de venir au tableau. […] Alors ils doivent calculer l'expression avec x = 1,254. Ces deux expressions sont semblables, mais les valeurs de x sont un peu différentes. […] Vous observez, s'il vous plait. 1,254, d'abord ils font, ils appuient Shift Sto A, c'est-à-dire stocker cette valeur de x dans A. Puis ils calculent le numérateur y qui est égale à B, puis stockent dans B […] Pourquoi doit-on mémoriser l'expression, pourquoi doit-on mémoriser x dans A ? La valeur 1,254 dans A ?
E : Pour ne pas devoir re-écrire la valeur 1,254.
P : Et si on l'avait pas fait. Qu'est-ce qu'on pourrait faire ?

[1] C'est Nguyen Chi Thanh

E : Oui, on fait avec Ans.
P : Mais dans ce cas, comment on calcule le dénominateur ?
Es (plusieurs en même temps) : On doit noter le résultat du numérateur.
P : Donc on doit noter le numérateur, n'est-ce pas ?
Es : Ouis.
P: Mais vous le savez, le groupe récepteur, ils peuvent pas, on ne peut pas écrire sur le papier. Alors, on doit enregistrer x dans une mémoire variable, A par exemple.

Ce que l'enseignant écrit au tableau

1,254 → A
Numérateur d'y : A × A × A × 2 + 15 × A – 5 → B
Dénominateur d'y : 32,1 × A – 27,7534 =
~~A ÷ Ans~~
B ÷ Ans

Dans ce bilan, la mémoire variable sert à stocker un nombre et diffère de la mémoire Ans : la mémoire variable est utilisée dans le calcul 3 afin de ne pas devoir noter le nombre M(v), résultat intermédiaire, sur papier.

Classes 2nd F

• *Premiers Bilans en groupe*
Avec les trois premiers groupes, l'enseignant choisit d'aborder dans le bilan essentiellement l'oubli des parenthèses. C'est le cas des programmes des binômes à qui il demande d'aller exécuter publiquement leur programme sur Alpro.
Nous allons présenter sur le bilan du dernier groupe. Le premier programme rendu public est un programme de calcul M_v, le second est un programme (A, B, Ans, Ans). Les programmes en actes des binômes invités au tableau sont donnés ci-après :

2nd F1 (groupe 1)	2nd F1 (groupe 2)	2nd F2 (groupe 3)	2nd F2 (groupe 4)
1,254 Sto A ; 2 × A × A × A +15 × A - 5 = ; ÷ 32,1 × A – 27,7534 ; Sto B ;	1,257 = 4 × Ans × Ans × Ans – 9 ×1,257 + 21 = ÷ 31,2 × 1,257 – 26,7184 = Ans × Ans – 7 × Ans + 9 =	1,254 = ; Ans × Ans × Ans = ; Ans × 2 ; Ans + 5 × 1,254 – 5 = ; Ans ÷ 32,1 × 1,254 – 27,7534 = ;	1,257 Sto A ; 4 × A × A × A - 9 × A +21 Sto B ; 31,2 × A – 26,7184 Sto C ; B ÷ C Sto A ; A × A – 7 × A + 9 = ;

Après la présentation publique du programme du groupe 4, l'enseignant va chercher à « améliorer » ce programme :

P : **Ca, ça peut-être un tout petit peu amélioré**. Ca, ça à la marge mais c'est simplement pour vous montrer. Léo, Léo. Simplement pour vous montrer qu'il y a une fonctionnalité, euh auprès de laquelle vous êtes totalement passés pour la plupart. Il y en a, en fait, j'ai entendu un ou deux qui en ont parlé, vous avez utilisé des mémoires A, B, C. Et il y a quelque chose que vous n'avez pas utilisé. La touche ?
E : Ans
P : La touche Ans. C'est-à-dire que peut-être, pour certains endroits là, au lieu d'utiliser une mémoire A, B ou C, vous auriez pu utiliser ?
Es : Ans
P : Ans, qui donne quoi ?
Es : La réponse
P : **La réponse du dernier calcul.** Alors, **à quel endroit on aurait pu l'utiliser**, par exemple ?
E (un autre) Bah, B divisé par…
P : Ouais, viens nous montrer.
E : (Elle veut pas aller au tableau) Mais non, je…
P : À quel endroit on peut utiliser par exemple.
E (un autre élève) : Pour le dénominateur.

P : Oui.
E : Et bah, on calcule mais on //Après on fait divisé par Ans.
P : On ne le mémorise pas. Et là, la place de C on met Ans. D'accord ? Et il y a un autre endroit éventuellement, qu'on peut utiliser ? Guillaume.
E : Dans A.
P : C'est donc à la fin, on n'est peut-être de remettre dans A et à la place de A et à la place on peut mettre Ans Ans, etc. C'est une touche ah. Il y a 3 lettres mais c'est une seule touche ah ? D'accord ? Voilà, est-ce qu'il y a d'autres questions sur la calculatrice, l'utilisation et sur des touches ?

Ce qu'écrit l'enseignant au tableau :

1er tableau	*2e tableau*
1,257 Sto A Numérateur Sto B = Dénominateur Sto C B / C Sto A A × A –	1,257 Sto A Numérateur Sto B = Dénominateur ~~Sto C~~ B / ~~C~~ Sto A Ans ~~A~~ × ~~A~~ … Ans Ans

Dans une séance en classe entière, l'enseignant a voulu conclure sur l'usage de mémoires variable et Ans et sur ce qui différencie ces deux types de mémoires :

Et puis, enfin, un petit mot sur l'usage de mémoire. Alors, je sais plus exactement ce qu'on a dit vendredi. Mais vous allez me, me redire. Donc, sur les calculatrices qu'on vous a proposées, il a quoi comme mémoires ?
Es : A, B, C
P : Il y avait A, B, C
E : Ans
P : Et Ans. Ca se comporte fondamentalement pareil ces trois, enfin ces deux types de mémoires ?
Es : Ouais.
E : Il y en a une…
P : Pierre
Pierre : **Enfin, il y en une qui s'efface petit à petit et**//
P : Donc, **Ans la mise à jour se fait comment** ? […] Mise à jour ? Eve, à quel moment pour la mémoire Ans ?
Eve: **Mettre au résultat**
P : C'est-à-dire, quelle touche ?
Eve : Égal
P : (Il écrit au tableau) Ah, mise **à jour à chaque fois qu'on appuie sur « égal », d'accord ? Donc, finalement la mémoire Ans, il faut un petit peu s'en méfier parce que, elle change, au fur et à mesure**. Euh, la mémoire A, Faustin, comment est-ce qu'on fait pour la mettre à jour, pour mettre une nouvelle valeur dedans ?

L'enseignant écrit au tableau :

A, B, C
Ans → mise à jour à chaque fois qu'on appuie sur =

Faustin : On tape la valeur.
P : Ouais, par exemple, 25
Faustin : Et Sto
P : Sto
Faustin : Et puis, après sur A
P : A, d'accord ? Et si on veut utiliser, la valeur de A ?
E (un autre) : On appuie sur A
P : On appuie sur A et on fait les calculs avec. Alors, il y a un autre moyen que vous n'avez pas du tout utilisé, c'est la touche Rcl. Vous faite A Rcl, Rcl qui signifie recall. Alors, qu'est ce qui se passe quand vous appuyez sur A Rcl, et bien, c'est la valeur qui est mémorisée apparaît à la place da A. En même temps, ça vous donne un contrôle sur la valeur que vous utilisez. […] Voilà des questions là-dessus.

L'enseignant a donc enseigné la « mise à jour », et la notion d'affectation des deux mémoires : « = » pour la mémoire Ans et « Sto » pour la mémoire variable.
Les discours de l'enseignant dans cette dernière synthèse et dans la phase de bilan pour le groupe 4 donne bien à la mémoire Ans le statut du « dernier résultat utilisé ».

Aucune mention dans les bilans des trois classes à la pratique « privée » de la touche « Ans~~Sto~~ », qui reste donc privée.

IV. Conclusion de l'analyse *a posteriori* de la situation 1

Les résultats de l'analyse des calculs routiniers 1 et 2 valident notre hypothèse de recherche selon laquelle, dans les calculs instrumentés, le rapport à la calculatrice dans EMS est essentiellement celui d'une machine arithmétique.

L'examen des fichiers historiques d'Alpro récupérés permet de reconstituer les programmes en actes des élèves. Ces programmes permettent de rendre visible d'autres résultats concernant les calculs instrumentés.

Tous d'abord le calcul instrumenté *dans l'institution française* a deux caractéristiques propres :

- Une règle du contrat didactique du calcul instrumenté, rendant acceptable comme résultat du calcul la valeur approchée, arrondie ou tronquée, du nombre affiché à l'écran de la calculatrice.
- Une pratique privée (Ans~~Sto~~) de découpage du calcul en sous calculs, découpage permis par le stockage *non intentionnel* du dernier résultat dans la mémoire Ans.

Dans *l'institution vietnamienne*, le vide didactique concernant les calculs instrumentés (chapitre C2) semble effacer les quelques connaissances instrumentales acquises au collège.

Rappelons que l'objectif de cette situation est aussi de faire évoluer le rapport instrumental à la calculatrice, en particulier en permettant l'identification des touches mémoires variables et Ans et leur différenciation.
La situation portant sur un calcul non routinier (calcul 3) permet non seulement de mettre en évidence cette évolution mais aussi d'initier une situation d'écriture, dans le langage évolué Alpro, d'un programme informatique à une machine à mémoire.

L'entrée dans cette nouvelle situation a permis à presque tous les élèves d'accéder aux deux types de mémoires. Seules trois élèves restent en échec.

Les programmes en actes font apparaître, chez certains élèves, une phase expérimentale, durant laquelle ils explorent les touches mémoires et débouchent sur leur usage pertinent.

L'exécution du programme écrit par autrui ainsi que les premières rencontres avec les mémoires dans les phases exploratoires des calculs 1 et 2 semblent avoir contribuer de façon cruciale au processus d'instrumentation des touches mémoires d'Alpro

Le stockage des nombres dans les mémoires (Ans et mémoires variables) devient intentionnel. C'est le résultat le plus probant concernant l'instrumentation des touches mémoires. De ce fait la touche Ans, mémoire la plus utilisée, devient une mémoire visible.

Mais les touches mémoires, que ce soit Ans ou les mémoires variables, ont, dans l'écriture des programmes, un statut de « variable mathématique » : une mémoire contient un seul nombre durant tout le calcul.

La caractéristique d'effaçabilité n'est pas encore mise en place pour la majorité des élèves, quelque soit leur niveau : Alpro n'est pas encore une machine à mémoire effaçable. Ce sera l'un des enjeux des situations suivantes de l'ingénierie didactique.

Nous allons maintenant dans des études de cas examiner de façon qualitative des processus d'instrumentation des mémoires plus ou moins aboutis.

Chapitre 1

Troisième partie
Étude du cas de trois binômes dans la situation 1

Introduction

Dans cette partie, nous allons essayer, à l'aide des fichiers historiques et des enregistrements audios des binômes, d'analyser le processus d'instrumentation des touches mémoires variables A, B, C et mémoire Ans.
Pour ce faire, nous choisissons trois binômes caractérisés par des parcours différents vis-à-vis de l'usage de l'instrumentation des mémoires.
Dans ce qui suit, nous avons grisé tout ce qui renvoyé par l'écran d'Alpro suite à des appuis de touches des élèves.
Nous nous intéresserons tout particulièrement aux touches mémoires de la calculatrice et des touches associées comme la touche « = » pour Ans, Sto et Rc pour les mémoires variables.
Mais l'instrumentation d'autres touches sont premières comme les touches de correction, « AC/ON » et « C/OFF ». Rappelons les fonctionnalités de ces deux touches :

- AC/**ON** : efface l'écran de la calculatrice, *mais pas un nombre mis dans les mémoires y compris un nombre-résultat mémoriser automatiquement dans Ans* ; Shift AC/**ON** : démarre la calculatrice ;
- C/**OFF** : efface un chiffre d'un nombre afficher à l'écran de la calculatrice, *sans toucher aux nombres mis en mémoire* ; Shift C/**OFF** : arrête la calculatrice.

I. Un rapport « machine arithmétique » à Alpro qui peut se maintenir

Nous choisissons de suivre un binôme de 2nd F1 dont les procédures de calculs restent attachées à la machine arithmétique. N'ayant pas d'enregistrements audio pour ce binôme, nous nous contentons ici de leurs réponses écrites et du suivi de l'exécution de leur calcul sur Alpro (fichier historique).

I.1. Les calculs 1 et 2

Les programmes en actes finaux pour les calculs 1 et 2 (reconstituées à partir des fichiers historiques) et les résultats écrits des deux élèves notés E1 et E2 sont donnés dans le tableau 1 ci-après :

E1	E2
Calcul 1 3,141 × 2 × 3,141 + 3,141 + 1 = ; **Calcul 2** 8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,141759682 × 3,141759682 - 3 × 3,141759682 - 1 =	**Calcul 1** 2 × 3,141 × 3,141 +3,141 + 1 = ; **Calcul 2** 8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,141759682 × 3,141759682 - 3 × 3,141759682 - 1 =
Résultat 1 : 23,8 **Résultat 2 :** 296,8 **Nombre d'appuis de touche :** 78	**Résultat 1 :** 23.87 **Résultat 2 :** 296.8 **Nombre d'appuis de touche :** 78

Tableau 1. Résultats écrits et programmes en actes finaux lors de calculs 1 et 2 pour les deux élèves de 2nd F1

Aucun des deux élèves n'utilise des mémoires dans sa procédure pour les calculs 1 et 2. L'examen de leurs fichiers historiques le confirme.

Voici les programmes en actes des calculs 1 et 2 de l'élève E1 :

Calcul 1:
[2 × 2 × 3,141+1= [13,564] C/OFF + 3,141 + 1 = [16,705] C/OFF(15 appuis)] *2 × 3,141 × 3,141 + 3,141 + 1 = [23,872762] ;*
Calcul 2 :
[AC/ON 8 × 3141759682 AC/ON] *8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,141759682 × 3,141759682 − 3 × 3,141759682 − 1 = [296,88424] ; AC/ON 8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,141759682 × 3,141759682 − 3 × 3,141759682 − 1 = [296,88424]*

Cet élève semble maîtriser les touches C/OFF et AC/ON : le gain (d'appuis de touches) apporté par AC/ON lui fait abandonner la touche C/OFF dans le calcul 2. Dans le calcul 1, pour effacer l'écran, il va jusqu'à appuyer 15 fois sur la touche C/OFF !
Il ne fait pratiquement pas de fautes de frappe, ce qui est remarquable pour le calcul 2, l'entrée de v étant très coûteuse. Conscient du risque d'erreur, il exécute deux fois le calcul instrumenté.

I.2. Phase de bilan à la fin du calcul 2

L'enseignant désigne un élève dont le nombre d'appuis est 64 et le programme de calcul en acte utilise la mémoire Ans :

3,141759682 = 8 × Ans × Ans × Ans = + 6 × 3,141759682 × 3,141759682 = - 3 × 3,141759682 = ;

Cet élève vient exécuter ce programme sur Alpro publique. L'enseignant ne fait aucun commentaire sur l'usage de cette mémoire.

I.3. Écriture d'un programme en langage des touches d'Alpro (phase 1, calcul 3)

Le binôme E1 et E2 ont à faire le calcul 3 suivant :

$z = y^2 - 5y + 7$ où $y = \dfrac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$ avec x = 1,254

Dans la phase 1 du calcul 3, E1 écrit le programme de calcul pendant que E2 l'exécute sur Alpro :

Programme écrit sans mémoire

2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 - 27,7534 × 2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 - 27,7534 − 5 × 2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 - 27,7534 + 7 = [-117,894067]

Programme en acte

2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 - 27,7534 × 2 × C/OFF × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 − 27,7534 − 5 × 2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 - 27,7534 + 7 = [-117,894067] ;

Le programme en acte est identique à celui écrit sur leur fiche de réponse, à part la correction C/OFF, ce qui montre le soin méticuleux avec lequel ces élèves entrent les données.

Quel est l'algorithme de calcul de ce programme ?
Le nombre $y = g(v) = \dfrac{2 \times v \times v \times v + 15 \times v - 5}{32{,}1 \times v - 27{,}7534}$ est remplacé directement dans $z = y^2 - 5y + 7$.
Ils contournent ainsi la nécessité du stockage de v dans un calcul coûteux en entrées de v 15 entrées).

Chaque y est « linéarisé » en « oubliant les parenthèses » : y = 2 × v × v × v + 15 × v – 5 ÷ 32,1 × v – 27,7534.

I.4. Rétroactions apportées par la phase de décodage (phase 2, calcul 3)

[À propos du calcul $z = y^2 - 7y + 9$ où $y = \frac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$ avec x = 1,257] le binôme reçoit le programme suivant :

31,2 × 1,257 – 26,7184 = Ans
4 × 1,257 × 1,257 × 1,257 – 9 × 1,257 + 21 = Ans

Ce programme ne fait appel à l'usage d'aucun stockage intentionnel dans une mémoire. Ainsi l'exécution de ce programme ne peut contribuer à l'instrumentation des touches mémoires pour le binôme E1 et E2.
Ci-après le programme en acte exécuté sur Alpro à partir du programme reçu et le message renvoyé par l'écran d'Alpro :

31,2 × 1,257 − 26,7184 = [12,500] Ans 4 × 1, 257 × 1,2571 × C/OFF(4 appuis) ,257 − 9 × 1,257 + 21= [Erreur, un nombre décimal comporte une seule '.']

Ce binôme, en voulant corriger une erreur d'entrée (4 appuis sur C/OFF), n'efface pas la virgule, et entre « 1,,257 ». Cette erreur d'entrée passe inaperçue.
Ils recopient l'écran : « Erreur, un nombre décimal comporte une seule '.' » sur la fiche navette de l'autre binôme.

I.5. Rectification du programme suite à la phase de décodage

L'exécution de leur programme par le binôme récepteur aboutit à un résultat à l'écran :

[-114, 894067]

Étant donné la pauvreté des rétroactions des deux ordres (exécution de leur programme et du programme d'autrui), il ne modifie pas leur programme initial et se contente de l'exécuter à nouveau :

2 × 1,24 AC/ON 2 × 1, 254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 − 27,7534 × 2 × 1,254 × 1,254 + C/OFF × 1,254 + 15 × 1,254 − 5 ÷ 32,1 × 1,254 − 27,7534 − 5 × 2 × 1,254 × 1,254 × 1,254 + 15 × 1,254 − 5 ÷ 321 × 1,254 − 27,7534 + 7 = [-117,7182726]

Ils trouvent un résultat du même ordre de grandeur que celui qu'ils ont obtenu dans la première phase (-117,7182726 et -117,894067) : cela les conforte dans la « validité » de leur programme.

I.6. Conclusion

Le rapport de ce binôme à la calculatrice Alpro reste donc lié à la machine arithmétique. Les seules touches qu'il semble maîtriser en dehors des touches d'entrée des nombres et des opérations sont les touches de corrections C/OFF et AC/ON.
Un faisceau de raisons contribue à cette résistance :

- Les élèves E1 et E2 ne connaissent pas les touches mémoires ;
- Dans la phase de bilan du calcul 2, l'enseignant ne fait aucun commentaire sur l'usage des mémoires ;
- Les rétroactions de la phase 2 sont quasi inexistantes : le programme qu'ils reçoivent pour exécuter le calcul 3 n'utilise pas intentionnellement de mémoires ;
- le programme qu'ils écrivent en langage Alpro aboutit bien à un résultat numérique qu'ils n'ont pas pas les moyens de mettre en cause ;
- leur algorithme de calcul est une réponse (erronée) au problème de la répétition.

II. De Alpro « machine arithmétique » à Alpro « machine à mémoire Ans »

Nous allons analyser le protocole d'un binôme de la classe de 1er V qui évolue d'un premier programme de calcul en acte (calcul 1), à une machine arithmétique à un programme de calcul écrit adressé à une machine à mémoire Ans.
L'observatrice (Ob.) de ce binôme est Annie Bessot.

II.1. Les calculs 1 et 2

L'élève E1 travaille sur sa calculatrice personnelle Fx570 MS alors que E2 travaille sur Alpro.

Nous donnons dans le tableau 2 les résultats écrits ainsi que, pour E2, les programmes en acte finaux pour les calculs 1 et 2.

E1	E2
Travail avec la calculatrice personnelle.	**Calcul 1** 2 × 3,141 × 3,141 +3,141 + 1 = ; **Calcul 2** 8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,141759682 × 3,141759682 - 3 × 3,141759682 - 1 = ;
Résultat 1 : 23,872762 **Résultat 2 :** 262,9403378 **Nombre d'appuis de touche** : 54	**Résultat 1 :** 23,872762 **Résultat 2 :** 296,888424 **Nombre d'appuis de touche :** 80

Tableau 2. Résultats écrits et programmes en actes finaux lors de calculs 1 et 2 pour les deux élèves de 2nd F1

Dans le calcul 2, les nombres n d'appuis de touche de E1 et de E2 (54 et 80) permettent d'affirmer, avec une forte probabilité, que ces deux élèves n'utilisent pas de mémoires dans leur calcul instrumenté.
L'examen du fichier historique de E2 confirme cette affirmation. E2 fait 3 fautes de frappe qu'il corrige avec C/OFF, pour ne pas avoir à tout effacer, et obtient le nombre à l'écran 296,888424 qu'il recopie sur sa feuille.

> 8 × 3,141759682 × 3,141759682 × 3,141759682 + 6 × 3,475982 C/OFF (6)[1] 14175962 × 3, 141759682 − 3 × 3, 141759682 − 1 = [296,8884228] C/OFF(30) 682 × 3,141759682 − 3 × 3,141759682 – 1 = [*296.888424*]

Le coût de la correction C/OFF « chiffre après chiffre » pour son entrée erronée de v, engage E2 dans une phase exploratoire de comparaison de l'usage des touches AC/ON et C/OFF, puis de la touche Ans.

> [C/OFF(80) AC/ON AC/ON C/OFF AC/ON(6) 9(18) × × = [Erreur Synt. Expression non Valide] AC/ON 9 × 9999 AC/ON × × = [Erreur Synt. Expression non Valide] AC/ON AC/ON] Ans ⊟ [Ans] [296.888424] AC/ON Ans AC/ON Ans = [Ans] [296.888424] AC/ON Ans = [Ans] [296.888424] AC/ON C/OFF Ans (5) = [Erreur Synt. Expression non Valide] AC/ON] *Ans × Ans = [Ans × Ans] [88142,73631] ÷ Ans = [Ans÷ Ans] [1]Ans = [Ans][1]Ans Ans AC/ON Ans = [Ans][1] Ans × AC/ON* ;

Le premier appui sur la touche Ans, précédé de la touche ⊟ fait apparaître d'abord le symbole Ans sur la ligne d'expression et le nombre 296,888424 sur la ligne de résultat, résultat déjà obtenu et écrit : c'est l'élément déclencheur du processus d'instrumentation de la touche Ans et de la touche AC/ON. Cette dernière touche efface l'écran, mais n'efface pas le nombre dans Ans ! Il fait ensuite plusieurs essais pour continuer son exploration et tester le fonctionnement de ces touches.

[1] C'est-à-dire que cet élève tape la touche « C/OFF » 30 fois de suite.

II.2. Phase de bilan à la fin du calcul 2

L'enseignant désigne un élève dont le nombre d'appuis est 32 et le programme de calcul en acte utilise la mémoire Ans :

3,141759682 = 8 × Ans × Ans × Ans + 6 × Ans × Ans - 3 × Ans - 1 = [*296.888424*]

La phase exploratoire de la touche Ans par E2 le rend réceptif à l'usage de Ans dans ce programme. Par contre, on ne peut rien dire pour E1, peut-être prend-elle connaissance pour la 1ère fois de l'usage de cette touche.

II.3. Écriture d'un programme en langage des touches d'Alpro (phase 1, calcul 3)

Le binôme E1 et E2 ont à faire le calcul 3 suivant :

$z = y^2 - 7y + 9$ où $y = \frac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$ avec x = 1,257

Le binôme a été observé et enregistré : nous donnons ci-après des extraits de leurs interactions en les commentant.

Écriture d'un programme à mémoire Ans, première tentative algébrique

E1 propose pour le calcul une factorisation du numérateur et l'usage de Ans.

11. E1 : Ah, bon ? **Attend, ou on met x en facteur. Alors le calcul est x multiplié par $4x^2 - x$, non $4x^2 - 9$**…Ou on prend quand même **1.257 comme Ans** ?

E2 commence à enseigner à E1 l'usage de la touche AC/ON (AC) et de la touche Ans.

12. **E2 : Non, voyons, 1.257. Écris le, 1.257 égale AC**.
13. E1 : Egale et AC ? (Elle tape sur Alpro)
14. E2 : **Oui, égale et ensuite AC pour que ça disparaisse**. Et puis encore AC, oui ça va comme ça.
15. E1 : Puis il faut ON ?
16. E2 : **Non, AC. Ca va, puis 4 multiplié par Ans, multiplié par Ans, par Ans.**

Ce que tape le binôme, à ce moment-là, synthétise de façon lumineuse l'instrumentation des touches Ans et AC/ON : entrée de v puis appui sur AC/ON pour nettoyer l'écran, puis appui sur Ans et la touche « = » pour vérifier le contenu de Ans. Cette vérification est suivie de l'usage de Ans comme une variable mathématique.

1,257 = AC/ON Ans = [1,257]
4 × Ans × Ans × Ans = [7,944486372] AC/ON

Le problème de l'indisponibilité des parenthèses et de l'interdit de la mémoire papier

Ce binôme prend conscience de l'indisponibilité des parenthèses et de l'interdit de noter le résultat sur le papier au moment de la division.

27. **E2 : Mais c'est la division, que doit-on faire** ?
28. E1 : Egale ceci et AC/ON n'est ce pas ? Et on continue de faire Ans pour ceci. Oui, AC/ON, n'est-ce pas ?
29. E2 : **Mais ici, il y a une division et nous on n'a pas de parenthèses, que doit-on faire ? C'est difficile.**
30. E1 : Ah, oui…**Mais et on ne doit pas écrire sur le papier pour enregistrer le résultat ah ?** Ah, oui, c'est possible, j'ai trouvé un moyen.

A la recherche d'une solution avec la mémoire Ans : les limites de Ans

Le stockage de la mémoire Ans se fait de manière invariante par appui sur la touche « = » suivi de l'appui de touche AC/On pour nettoyer l'écran et continuer le calcul.

E1 oublie la touche « = » pour ne retenir que la touche AC/ON pour le stockage d'un nombre dans Ans !

32. E1 : On va faire comme suivant. Ce x égale Ans, n'est ce pas ?
33. E2 : Mais n'oublie pas que cette valeur est seulement celle d'y. Alors que z c'est y^2 moins 7 y…
34. E1 : Oui, je sais, mais doucement. Ceci est égal à Ans, n'est-ce pas ? Et on peut mémoriser le résultat par AC/ON, c'est d'accord ? **AC/ON et on mémorise ce résultat.** Et maintenant c'est…

Ils rencontrent les limites de Ans : il leur faudrait deux mémoires Ans, une pour le numérateur, une pour le dénominateur ! Ils cherchent alors une réécriture algébrique du quotient qui leur permette d'éviter la division.

35. E2 : On va factoriser z, notre z est égal $y^2 – 7y + 9$ c'est…
36. E1 : Mais on a seulement 10 minutes. Ca sert à quoi de diviser par ce bloc ?
37. E2 : 3 et 34…1936. Oh, là là, il n'y a pas de solution.
38. E1 : Alors, si on fait Ans ceci, on considère que ceci est Ans. Donc c'est fait. Mais alors le dessous ? Mais qu'est ce qu'on fait avec le dessous ?
39. E2 : C'est impossible de le mettre en produit.
40. E1 : Oui, quoi, maintenant on cherche à trouver la solution avec ce bloc ? Car si on le mémorise, on ne sait pas comment on calcule le dessous ?
41. E2 : Ouais. On calcule celui-ci, n'est pas difficile, celui-ci aussi pas difficile, **mais le problème est comment diviser celui-ci par celui-ci ?**
42. E1 : Ouais.
43. Ppu : Attention, il faut que vous notiez votre nom et prénom sur les feuilles.
44. E2 : **Tiens, par exemple, si on fait Ans encore une fois, ce sera celui-ci lui-même et non pas celui-ci.**
45. E1 : Oui et on peut diviser par ce bloc là mais on ne peut pas diviser par le produit de ces deux là. Le Ans-là, est-ce qu'il garder ça ? […]
60. E1 : Si, on le calcule maintenant. On vient de le faire, c'est ça ? Allez, calcule le car maintenant il suffit de lui demander de diviser une seule…valeur du dessous. Alors, fais le.
61. E2 : Non, ce n'est pas possible. C'est facile de le comprendre entre nous, mais ce n'est pas possible […]

Le programme en acte à ce moment-là.

31,2 × 1,257 – 26,7184 = [12,5] Ans – 5 = 1,257 = AC/ON Ans = 4 × Ans × Ans × Ans - 9 × Ans + 21 = [17,63141837] ÷ 12,5 = [1,41051891] AC/ON

Une solution pas vraiment satisfaisante

Ils ne trouvent pas de solution et sont contraints par l'interdit de la mémoire papier.
Le nombre M(v) leur est donné par le binôme du groupe d'à côté : 12,5.
L'urgence (il faut rendre la feuille) et la simplicité du nombre M(v) les poussent à l'écrire dans leur programme, tout étant conscient qu'ainsi ils violent deux interdits, celui de la mémoire papier et celui de ne pas avoir trouvé eux-mêmes le résultat !

64. E1 : Ouais (elle demande au groupe à côté) T'as déjà calculé le dessous ? C'est combien ?
65. E (du groupe à côté) : 12.5
66. E2 : 12.5 ! Oh, ce nombre est beau, ce résultat !
67. E1 : Ok, maintenant, on prend celui-là divisé par 12,5
68. E2 : **Non, on ne peut pas faire comme ça. D'où on trouve le nombre 12,5**.
69. E1 : Oui, mais on le sait. Pourquoi pas ?
70. E2 : Oui, c'est OK, ça fait rien.
71. E1 : N'est ce pas. En tout cas, il le saura.
72. E2 (Il lit les instructions puis E1 les tape sur Alpro) : Multiplier par Ans, Multiplier par Ans, Multiplier par Ans puis moins 9 Ans plus 21. Pourquoi ils peuvent appuyer comme ça ? Egale.
73. E1 : Egale et divise par ?
74. E2 **: Oui, tu peux l'essayer. Egale. Puis divise par 12,5**. **Non, non, il faut un égal** […]
75. E1 : Plus 9. Tu vois. Ce n'est pas faux. On l'espère en tout cas. On peut diviser par ce résultat ? **Faire comme ça ? Personne ne l'interdit, n'est-ce pas ?**
76. E2 : Si l'on ne suit pas cette méthode alors on …
77. E1 : N'est-ce pas. Ici on peut utiliser du papier, de stylo et la calculatrice. **Mais est-ce qu'on peut diviser en utilisant un résultat ?** Alors, essayons de trouver un autre moyen.

L'instrumentation de la mise en mémoire Ans continue pour E1 : E2 réaffirme à plusieurs reprise la nécessité de la touche « = ».

Le programme en acte

Ans = 4 × Ans × Ans × Ans - 9 × Ans + 21 = [17,63141837] ÷ 12,5 = [1,41051891] AC/ON Ans × Ans – 7 × Ans + 9 = [1,115931226]

Le programme écrit à la suite de ce programme en acte

1,257 = AC/ON
4 × Ans × Ans × Ans – 9 × Ans + 21 = ÷ 12.5 = AC/ON
Ans × Ans – 7 × Ans + 9 =

Ils vérifient ce programme de calcul trois fois sur Alpro et aussi sur la calculatrice personnelle de E1 : ils retrouvent le même résultat (1,115931226).
Ce binôme a ressenti le besoin d'une autre mémoire … Ans.
Leurs interactions montrent qu'ils ne connaissent, ni l'un ni l'autre, l'existence des touches de mémoires variables qui leur aurait permis de sortir de l'impasse. Ils donnent une solution qui ne les satisfait pas.

II.4. Rétroactions apportées par la phase de décodage (phase 2, calcul 3)

[À propos du calcul $z = y^2 – 5y + 7$ où $y = \frac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$ avec x = 1,254], le programme reçu pour être exécuter par notre binôme est un programme proche du leur, puisque ce binôme propose deux mémoires Ans, l'une notée (1), l'autre (2).

1,254 = AC/ON
2 × Ans × Ans × Ans + 15 × Ans - 5 = (1) AC/ON
3,12 × 1,254 – 27,7534 = (2) AC/ON
(1)/Ans = (3) AC/ON
z = Ans × Ans – 5 × Ans + 7

Que font les élèves pour exécuter un tel programme qu'ils ont envisagé sans l'écrire ?

1,257 = 2 × Ans × Ans × Ans + 15 × Ans - 5 = [17,82724319] 3,12 × 1,254 – 27,7534 = [-23,84092] Ans Ans [Erreur Synt. Expression non Valide] AC/ON 1,254 = 2 × Ans × Ans × Ans + 15 × Ans - 5 = [17,75387013] 3,12 × 1,254 – 27,7534 = [-23,84052] ÷ AC/ON ÷ Ans = [Erreur Synt. Expression non Valide] AC/ON AC/ON ÷ Ans = ÷ [Erreur Synt. Expression non Valide]

Tout d'abord notons que la touche AC/ON n'est pas utilisée ici pour entrer un nombre dans la mémoire Ans : la touche « = » suffit.
Le binôme suit le programme à la lettre. En exécutant le programme, E1 remarque (128) :

> Pourquoi ils peuvent mettre Ans plusieurs fois comme ça ? Ce Ans ne peut pas mémoriser plusieurs résultats, n'est-ce pas ?

Ce commentaire montre que, pour E1, Ans ne peut contenir qu'un résultat.
Ils recopient le dernier écran : « Erreur Synt. Expression non Valide ».

II.5. Rectification du programme suite à la phase de décodage

Le programme qu'ils ont eu à exécuter a confirmé que l'on ne pouvait pas utiliser plusieurs mémoires Ans.

190. E2 : Comment, comment ? 1.257, égale puis AC/ON, et puis après, 4 multiplié par Ans, 4 multiplié par Ans, multiplié par Ans et moins 9 multiplié par Ans plus 21 et égale. Puis AC/ON. **C'est AC/ON qui a donc gardé ce résultat**. Et on obtient, Ans divisé par…Est ce que c'est cette somme divisée par ce produit ? N'est-ce pas ?
191. E2 : Ouais, ouais.
192. E1 : Donc est ce que c'est égal à la somme de chaque terme divisé par ce produit ?
193. E2 : Non, tu rêves ou quoi ? Ceci est divisé par ceci plus ceci divisé par ceci.

Il semble que si la touche « = » permet d'entrer le nombre v dans Ans, la touche AC/ON permet de le garder, alors qu'il n'est plus visible.
E1 et E2 cherchent toujours une réécriture du quotient qui leur permette d'éviter la division.

194. E1 (au brouillon) : Non, c'est ce que A divisé par A moins B est égale à A, c'est-à-dire égale à A multiplié par 1 sur A moins 1 sur B ?

E1 écrit au brouillon : A : (A – B) $A\left(\frac{1}{A}-\frac{1}{B}\right)$.

Puis en désespoir de cause, ils explorent d'autres touches d'Alpro dont les touches mémoires :

AC/ON 9(34) C A C A AC/ON 9(91) (3) 8(9) 20 =

Ils ne rectifient pas leur programme.

II.6. Conclusion

Les deux élèves ont pu instrumenter la mémoire Ans lors de ces trois calculs ainsi que les touche « = », AC/ON, C/OFF, E2 dès le calcul 2.
L'entrée d'un nombre dans Ans se fait en tapant les chiffres du nombre, suivi des appuis sur les touches « = » [pour le mettre dans la mémoire Ans] suivi de l'appui sur la touche AC/ON.
Le rôle de la touche AC/ON pour ces élèves semble être de garder un nombre en mémoire, une fois effacé.

Ce binôme a ressenti le besoin d'une seconde mémoire pour résoudre le problème posé par la division. Les touches mémoires leur étant totalement inconnues, il leur aurait fallu une seconde mémoire Ans.

Ils ont eu recours à un subterfuge, manière détournée d'utiliser la mémoire papier, pour leur programme écrit utilisant uniquement la mémoire Ans. Ce programme est ressenti comme un échec.
Le seul recours envisagé est une solution « algébrique » qui transforme une division impossible en une multiplication. Cette tentative de récriture échoue aussi.

On peut penser alors que ces élèves sont prêts à découvrir les touches mémoires variables qui leur apporteraient une solution pour surmonter leurs échecs successifs.
Nous continuerons à suivre ce binôme dans la suite de l'ingénierie.

III. De la machine arithmétique à la machine à mémoires A, B, C et Ans (mémoires variables mathématiques) : une évolution suscitée

Nous allons analyser le protocole d'un binôme de la classe de 2nd F2 qui évolue d'un premier programme de calcul en acte (calcul 1), à une machine arithmétique à un programme de calcul écrit adressé à une machine à mémoire A, B, C et Ans.
Ce binôme a été observé par Annie Bessot (Ob.).

III.1. Les calculs 1 et 2

Les programmes en actes finaux pour les calculs 1 et 2 (reconstituées à partir des fichiers historiques) et les résultats écrits sont donnés dans le tableau 3 ci-après :

E1	E2
Calcul 1 2 × 3,141 × 3,141 + 3,141+1 = **Calcul 2** 3,141759682 = Ans × Ans × Ans = Ans × 8 = + 6 × 3,141759682 ;	**Calcul 1** 2 × 3,141 = Ans × 3,141 = Ans +3,141 = Ans + 1= ; **Calcul 2** 3,141759682 = Ans × Ans × Ans × 8 = + 6 × 3,141759682 × 3,141759682 = - 3 × 3,141759682 = - 1 = ;
Réponse 1 : 23,872762 ; **Réponse 2 :** **Nombre d'appuis de touche :**	**Réponse 1 :** 23,872762 ; **Réponse 2 :** 296,888424 ; **Nombre d'appuis de touche** : 45 ;

Tableau 3. Résultats écrits et programmes en actes finaux lors de calculs 1 et 2 pour les deux élèves de 2nd F2

Le cas de E1 : exploration non aboutie de la mémoire variable A

L'élève E1 n'a donné aucun résultat pour le calcul 2. À partir des fichiers historiques, essayons de reconstituer ce qui l'a conduit à cette absence de réponse.
Pour le calcul 1, l'élève E1 n'utilise pas de mémoire, corrige d'abord avec la touche C/OFF puis avec la touche AC/ON. Il semble donc maîtriser l'usage des ces deux touches pour effacer un nombre.

[2 × 3,141 C/OFF (8) AC/ON] 2 × 3,141 × 3,141 + 3,141 + 1 = [23,872762];

Pour le calcul 2, E1 entre dans une longue phase exploratoire des touches mémoires variables qui n'aboutit pas. Il sait que la touche A est associée à une mémoire et il tente d'utiliser les touches associées Sto, Rcl et « = », mais les messages renvoyés par l'écran d'Alpro n'apportent pas de réponse à la question : comment stocker un nombre dans une mémoire variable ? Il tape même A × A × A × 8 + 6 × A × A – 3 × A – 1, mais sans avoir réussi à stocker dans A le nombre V

[A = [Erreur. Mémoire non initialisée] AC/ON 3,141759682 A A STO AC/ON 3,141759682 A = [Erreur Synt. Expression non Valide] AC/ON A 1. AC/ON = AC/ON AC/ON AC/ON AC/ON 3,141759682 × 3,1417 AC/ON A 3 AC/ON B A C 5 = [Erreur Synt. Expression non Valide] AC/ON = AC/ON AC/ON AC/ON 3 = AC/ON Shift AC/ON 9 Shift Shift Shift Sto Rcl Sto Sto 789 Sto = AC/ON 3,141759682 = A AC/ON A × A + AC/ON A × A × 8 + 6 × A × A – 3 × A – 1 = [Erreur. Mémoire non initialisée] AC/ON

L'apparition de Ans × 8 puis de Ans + 6 × 3.141759682 × 13Ans à l'écran lui fait abandonner son exploration de la mémoire variable A pour celle de la touche Ans.

3,141759682 × 3,141759682 = [9,870653899] × 8 [Ans × 8] = [78,9652312] + 6 × 3,141759682 × 13 Ans [Ans+6×3.141759682×13Ans] AC/ON 9 AC/ON 1,141 AC/ON 8 × 3,1416 AC/ON 3 × AC/ON] *3,141759682 = Ans × Ans × Ans = [Ans × Ans × Ans] [31.01122246] Ans × 8 =* [Ans × 8] [248.0897796] + 6 × 14 C/OFF C/OFF 3,11 C/OF 416 C/OF 759682 AC/ON ;

L'usage de la mémoire Ans s'appuie d'abord sur un découpage du calcul 2, typique de la stratégie Ans~~Sto~~. Il n'y a donc pas de stockage intentionnel d'un nombre dans la mémoire Ans. À la fin, E1 tape « 3,141759682 = Ans × Ans × Ans mais l'appui sur la touche « = » qui suit remplace le contenu de Ans. Le fonctionnement de Ans comme mémoire n'est pas encore stabilisé.
Le temps pris à l'exploration de la mémoire variable A n'a pas permis à cet élève de terminer le calcul 2.

Le cas de E2 : la pratique Ans~~Sto~~ s'oppose à l'émergence de la mémoire Ans

Pour le calcul 1, E2 découpe le calcul $2x^2 + x + 1$ en sous calculs : 2 × x × x + x + 1 = (((2 × x) × x) + x) + 1 à l'aide de la mémoire Ans~~Sto~~. Ce découpage fait apparaître plusieurs fois sur l'écran le mot « Ans ».

Le calcul 2 a t-il permis une évolution du stockage non intentionnel dans Ans (Ans~~Sto)~~ vers un stockage intentionnel (mémoire Ans) ?
Examinons le programme en actes, reconstitué à partir son fichier historique :

[AC/ON 3,141759682 × AC/ON 3,141759682 ⊟ Ans × Ans ⊟ [9,870653899] Ans × 3,141759682 = [31,01122246] Ans × 8 = [2480897796] AC/ON 3,141759682 × 3,141759682 × 3,141759682 = [31,01122246] Ans × 8 = [248,0897796]

L'appui sur la 1ère touche ⊟ permet le stockage du nombre v (3,141759682) dans la mémoire Ans, mais l'appui sur la 2e touche ⊟ l'efface pour stocker le « dernier résultat » 9,870653899. Le résultat auquel il aboutit (2480897796) lui renvoie une rétroaction négative par rapport à son usage de Ans : la pratique de découpage du calcul 1 s'oppose à un stockage stable du nombre v.

A = [Erreur. Mémoire non initialisée] AC/ON] 3,141759682 = Ans × Ans × Ans = [31.01122246] Ans × 8 = [248,0897796] A C/OFF Ans + 6 × 3,141759682 × 3,141759682 = [Ans+6×3.141759682×3.141759682][307,313703] Ans − 3 × 3,141759682 = [297,888424] Ans – 1 = [296,888424]

E2 n'arrivant pas à contrôler le contenu de Ans se résigne t-il à entrer v dans la suite du calcul.
E2 tente comme E1 d'utiliser les mémoires variables à deux reprises, pendant son calcul instrumenté par Ans, puis après avoir terminé son calcul :

[AC/ON B = AC/ON(3) A 6 = C/OFF(3) C/OFF C/OFF(3) A A C/OFF C/OFF];

Dans cette dernière exploration, lui aussi (comme E1) cherche comment stocker un nombre dans ces mémoires : il essaie « = ».
Comment leur connaissance de touches mémoires Ans ou variables évolue-t-elle après la phase bilan de l'enseignant à la fin du calcul 2 ?

III.2. Phase de bilan à la fin du calcul 2

L'enseignant a écrit au tableau

Réponse 1 = 23,872762
Réponse 2 = 296,888424
≈ 60 touches
64 touches

L'enseignant désigne un élève dont le nombre d'appuis est 62 et le programme de calcul en acte utilise la mémoire Ans :

3,141759682 = ; Ans × Ans × Ans = ; × 8 + 6 × 3,141759682 × 3,141759682 – 3,141759682 – 1 = ;

Ce programme en acte débute par un stockage intentionnel de v dans Ans, suivi d'un fonctionnement Ans~~Sto~~.
L'information qu'il apporte aux questions des élèves révélées par les phases exploratoires est donc le stockage intentionnel de v.

III.3. Écriture d'un programme en langage des touches d'Alpro (phase 1, calcul 3)

Le binôme E1 et E2 ont à faire le calcul 3 suivant :

$z = y^2 - 7y + 9$ où $y = \dfrac{4x^3 - 9x + 21}{31{,}2x - 26{,}7184}$ avec x = 1,257

Le binôme a été observé et enregistré : nous donnons ci-après des extraits de leurs interactions en les commentant.

Le stockage intentionnel de v dans Ans

Ce binôme s'engage dans un stockage intentionnel de v dans la mémoire Ans :

56. E2 : Ca sert à quoi ? Il veut nous embrouiller là. Ca sert à quelque chose. Vas-y. On va déjà calculer et après on divise par ça. On va d'abord calculer ces deux lignes et après on va calculer cette ligne.
57. E1 : Ouais, ouais, **je sais.** 1, 257.
58. **E2 : 3 fois. Ans fois Ans fois Ans et fois 4 moins 9 fois Ans plus…**
59. E1 : Plus 21. Divisé par, 31 virgule 2…

Le programme en acte qui accompagne cette interaction est le suivant

1,257 = Ans×Ans×Ans×4−9×Ans+21 = [17,63148637] Ans ÷ 31,2 × 1,257 – 6,7184 = [-6,008054539]
AC/ON 31,2 × 1,257 – 26,7184 = [12,5]

Le problème posé par l'indisponibilité des parenthèses

L'expression entrée « 4 × Ans × Ans × Ans - 9 × Ans + 21 ÷ 31,2 » est rendu visible à l'écran. E2 se rend compte de l'erreur de linéarité et tente de l'expliquer à E2 :

60. E2 : Ca c'est pas bon par ce que ça va faire ça divisé par 31 virgule 2 puis après ça va faire fois Ans moins 26. Tu vois ce que je veux dire ?
61. E1 : Non.
62. E2 : Ca va juste diviser ça par 31 virgule 2. Et après ça va multiplier par ça.
63. **E1 : Et il faudra des parenthèses//**

Première solution : mémoire papier

La première solution envisagée est de noter le nombre N(v) sur la fiche navette.

64. E2 : Non, non.
65. E1 : Tu veux qu'on fasse ? Comme tu dis, d'accord ? Vas-y, essaie.
66. E2 : Fois 1 virgule...
67. E1 : Moins.
68. E2 : Moins 26 virgule 7184. Moins ?
69. E1 : Moins 6.
70. **E2 : Et après on marque ce que j'ai dit ? On fait 31 virgule fois 1 virgule 257 moins 26 virgule 7184 égale. C'est 12,5. Alors c'est 17 virgule, 37 divisé.** Mais on ne trouve pas pareil là. On trouve pas pareil. Mais on prend celui là.

Le programme en acte qui accompagne cette interaction est le suivant

1,257 = Ans × Ans × Ans × 4 − 9 × Ans + 21 = [17,63148637] 31,2 × 1,257 – 26,7184 = [12,5]

La mémoire Ans contient le nombre 12,5 et elle est réutilisée dans le calcul. L'instrumentation de la touche Ans comme mémoire a évolué puisque les élèves savent quand l'utiliser.

17.63148637 ÷ Ans = [1,41051891];

Le rappel par l'enseignant de l'interdit de la mémoire papier

L'enseignant intervient pour leur rappeler l'interdit de la mémoire papier.

78. P : Donc **le message que vous devez passer c'est quoi ? C'est le résultat ?**
79. E2 : Bah, ouais.
80. P : Quel est l'intérêt de passer le résultat ?
81. E2 : Non, il faut qu'on passe toutes les touches pour arriver au résultat.
82. P : Voilà.
83. E2 : Il y en a un paquet ah ? […]

Suite à l'intervention de l'enseignant, les élèves se retrouvent de nouveau devant le problème de la division en l'absence de parenthèses. La seule réponse de E1 est de noter le résultat sur la fiche alors que E2 se tourne à nouveau vers les mémoires A, B, C qu'il ne sait pas initialiser

105. E2 : Tu tapes le nombre direct. Egale et après tu fais un point…

106. E1 : Point virgule là ?
107. E2 : Je ne sais pas. Tu mets un espace. Et puis Ans (…) Et après égale.
108. E1 : Egale Ans ?
109. E2 : Non, non. Après on va faire 31 virgule 2. On va faire ça divisé par Ans.
110. E1 : Il ne faut pas mettre le égal alors.
111. E2 : Non, après tu mets ça divisé. On va faire ça divisé par Ans.
112. E1 : Alors c'est Ans divisé par ça ?
113. E2 : **On peut pas car là, il y a une multiplication. Tu ne peux pas.**
114. E1 : Comment tu fais ? Il faut noter le nombre.
115. E2 : Mais eux, ils peuvent pas noter le nombre. Ils ont pas de…ils ont pas de stylo.
116. E1 : C'est merveilleux.
117. E2 : C'est chien.
118. E1 : **Y a un truc à écrire avec les mémoires, mais on sait pas utiliser…Il faut que je marque ce que tu marques ?**
119. E2 : On trouve 31 virgule. Ce n'est pas ça. Ils n'ont pas trouvé ça.
120. E1 : Mais il faut qu'on le note le machin. **On ne peut pas les mettre en mémoire ?**
121. E2 : **Mais on peut pas. Eux ils ne peuvent pas écrire.**
122. P : Ils ne peuvent pas écrire ?
123. E1 : **Puisque on a trouvé ces nombres. On les a marqués à côté puis on a refait un calcul.**
124. P : Mais ça, ça c'est votre brouillon et ensuite vous donnez ce qu'il y a à taper.[…]

Le programme qu'ils écrivent est limité au calcul du numérateur :

1.257 = Ans × Ans × Ans × 4 – 9 × 1.257 + 21 =

Ce qu'ils ont écrit sur leur brouillon :

17,6314837 ; - 6 ; y = 1,41051891 ; z = 1,15931225

Le programme en acte, reconstitué à partir de fichier historique, qu'ils ont tapé après l'écriture du programme est le suivant :

1.257 = Ans × Ans × Ans × 4 – 9 × 1.257 + 21 = ; 31,2 × 1,257 – 26,7184 ÷ Ans

Pour répondre au problème de la division en l'absence de parenthèse, sans noter de résultat,, ils ont calculé d'abord N(x), puis M(x) puis effectué la division M(x)÷N(x) (au lieu de N(x)÷M(x)). Cette solution n'a pas été écrite, ce qui montre qu'elle n'est pas vraiment acceptable pour les deux élèves.

III.4. Rétroactions apportées par la phase de décodage (phase 2, calcul 3)

Dans cette phase, ils reçoivent un programme contenant comme unique mémoire Ans. Pour cela, le programme ne respecte pas l'interdit de la mémoire papier : le résultat du calcul du numérateur, 17,7537013, est donné dans la fiche.

1) 1,254 = Ans × Ans × Ans × 2 + 15 × 1,254 − 5 =

2) 32,1 × 1,254 − 27,7534 =

3) $\frac{17,75387013}{Ans}$ =

4) Ans × Ans − 5 × Ans + 7 = z

Ils l'exécutent correctement ce programme et écrivent sur la fiche navette le dernier écran obtenu : 1,42030691.

III.5. Rectification du programme suite à la phase de décodage

Le programme qu'ils ont eu à exécuter est similaire à celui qu'ils ont écrit et la solution dans ce programme est le non respect de l'interdit mémoire papier. Ce programme n'a donc pas apporté de solution à leur problème.

Une situation didactique dont l'enjeu est le stockage d'un nombre dans une mémoire variable

L'observatrice, ayant suivi les vaines explorations des élèves des mémoires variables, intervient pour répondre à la question qu'ils se sont posée : comment stocker un nombre dans une mémoire variable ?

89. Ob. : Vous avez vu le problème ? Les mémoires, il faut stocker.
90. E2 : Ouais.
91. Ob. : Vous n'avez pas une touche qui est presque stocker ?
92. E2 : Oui. Mais il faut faire comment avec ces touches ?
93. Ob. : Vous essayez. Soit vous mettez le nombre et vous mettez Sto A, soit ...

Ils effectuent une combinatoire sur l'ordre de frappe des touches « v », « Sto » et « A », la validation de leurs essais étant le message envoyé à l'écran d'Alpro : « Stocké ». Nous ne donnons que la fin de cette exploration combinatoire :

Sto C/Off C/Off AC/ON A Sto 1, C/Off 1,2 C/Off C/Off C/Off Sto 1,254 C/Off C/Off C/Off 1,254 Sto A [Stocké]; AC/ON A × 2 = [2,508]

L'observateur, en position d'enseignant, achève cette phase didactique en institutionnalisant non seulement l'instruction d'initialisation de A mais en introduisant la touche Rcl. Puis elle demande aux élèves de poursuivre seul le travail de rectification de leur programme.

159. Ob. : Voilà. Vous allez mettre la valeur dans A. Vous tapez. Non. Vous tapez d'abord la valeur et puis vous mettez Sto A.
160. E2 : Oui d'accord.
161. Ob. Essaie avec un petit nombre pour voir si ça marche. Si vous voulez savoir ce qui dans la mémoire…Sto… Vous tapez Rcl. Vous avez jamais utilisé ça dans votre calculatrice ?
162. E2 : Non.
163. Ob. : Ca y est dans votre calculatrice ah ?
164. E2 : Ouais, ouais mais on l'a jamais utilisé.
165. E1 : Et après quand on met A c'est le chiffre qu'on a mis.
166. Ob. : Voilà.
167. E1 : Et fois 2 par exemple.
168. Ob. : Voilà et il contient, il contient le nombre. Bon, voilà, je ne parle plus ah. Débrouillez vous. Parce que, comme vous avez été tracassé avec ce problème depuis le début…

L'écriture d'un programme (A, B, Ans, C)

Les rôles sont bien répartis : E1 écrit pendant que E2 dicte ce qu'il tape sur Alpro.

169. E2 : Ouis, ça me perturbe. Mais. Et si on récrivait un message et on mettait en mémoire des trucs ?
170. E1 : Ouais.
171. E2 : Mais c'est super. Vas-y marque c'est bon.
172. E1 : C'est bon ? On refait tout.
173. E2 : Mais donne moi le brouillon et après je recopie. Ca fait…
174. E1 : Tu fais quoi là ? Dis moi avant que tu fasses le stockage ah ?
175. E2 : J'ai fait le stockage.
176. E1 : Mais bah il faut que je marque moi.
177. E2 : **Stocker. Tu fais 1 point 257 Sto A. Et après tu mets ON, AC/ON un truc comme ça et tu fais 4 fois A fois A fois A moins 9 fois A…**
178. P : Voilà, vous avez cinq, cinq gros minutes pour réviser votre message, le corriger. Vous ne le redonnez pas au voisin, ah. Allez.
179. E2 : Plus 21 et égale et tu fais Ans Sto B.

Au lieu de faire « Sto B », le binôme stocke le contenu de Ans (dans la mémoire B : « = Ans Sto B »). Cela montre un véritable contrôle par E2 du contenu de la mémoire Ans confirmé dans la suite des interactions.

Le programme en acte tapé par E2 durant l'interaction précédente

A Sto 1,25 AC/ON 1,257 Sto A [→ 1,257] [Stocké] AC/ON 4×A×A×A−9×A+21 = [17.63148637] Ans Sto B [Stoké] 31.2×A−26.7184 = [12,5]

Le problème de la division en l'absence de parenthèses ne pose alors plus de problème au binôme, même si E1 propose de mettre M(V) dans la mémoire C et E2 dans la mémoire Ans qu'il sait contenir le dernier résultat M(v). Le « tu fais rien » de E2 (189) signifie « tu n'écrit pas « = », car l'appui sur « = » change le contenu de Ans que je veux utiliser pour la division ».
Le stockage dans une mémoire *au cours du calcul* passe pour E2 par le stockage du contenu de la mémoire Ans, qu'il connaît.

180. E1 : Ouais.
181. E2 : Ca c'est le 1er calcul. Tu marques 1 et tu fais 2.
182. E1 : Ouais. 31 virgule 2. Et tu vas mettre en C ça ?
183. E2 : 31 virgule 2.
184. E1 : Et tu vas mettre en C ?
185. E2 (il rit) : 31 point 2 fois A moins 26 point 7184 est égale, est égale. 12 virgule 5. Et après tu mets B.
186. E1 : Tu n'as pas stocké ?
187. E2 : Non, oui j'ai stocké. C'est bon. B divisé par…
188. E1 : Egale 12 virgule 5. Je fais quoi ? Je fais Sto ? 12 virgule 5 ? Egale ?
189. E2 : Oui, ah non, non. Tu ne fais pas ça. Tu fais rien. **Après tu fais 4 et puis B divisé par Ans.**
190. E1 : Ouais.
191. E2 : Egale et après on stocke ça. Ca c'est y. Donc. **Egale. Ans Sto C […]**

Le programme (A, B, Ans, C) final

1) 1.257 Sto A AC ;
2) 4 × A × A × A − 9 × A + 21 = Ans Sto B ;
3) 31.2 × A − 26.7184 = ; B ÷ Ans = ; Ans Sto C ;
4) C × C − 7 × C + 9 = ;

Ils structurent leur programme à la manière du programme qu'ils ont eu à exécuter lors de la phase 2.
Dans ce programme chacune des mémoires variables A, B, C contient un seul nombre et a donc un statut de variable mathématique.

III.6. Conclusion

Le calcul 2 a montré que la pratique de découpage d'un calcul en sous calculs, apprise au collège via la mémoire Ans~~Sto~~, a permis de poser à ces élèves la question du contenu de la mémoire Ans, préalable au stockage intentionnel d'un nombre dans Ans. Le bilan à l'issue du calcul 2 ainsi que l'écriture d'un programme pour le calcul 3 leur permet de répondre à cette question.

Tout au long des calculs instrumentés par Alpro, les deux élèves de ce binôme ont exploré les touches mémoires variables qu'ils ne connaissent pas. Les messages envoyés à l'écran par Alpro ont donné à ces touches un statut de mémoire sans donner de réponse à la question du stockage d'un nombre dans cette mémoire.

Il a fallu que l'observateur organise une situation didactique pour que les élèves apprennent à utiliser la touche Sto de façon pertinente.
Les élèves se sont d'autant mieux saisis de cet enseignement qu'ils étaient dans une situation d'échec vis-à-vis du problème de l'écriture d'un programme concernant le calcul de l'image de v par une fonction rationnelle *sans la disponibilité des parenthèses*.

IV. Conclusion des études de cas

Ces études de cas confirment que le processus d'instrumentation des mémoires variables A, B, C s'avère problématique. L'intervention didactique de l'enseignant (qui peut être l'observateur) et l'exécution du programme à mémoire variable écrit par autrui semblent des conditions nécessaires pour initier un processus d'apprentissage de ce mémoires.

L'instrumentation de la touche mémoire Ans se montre moins problématique, l'usage de la mémoire Ans~~Sto~~, pratique « privée » chez des élèves sortant du collège en France, permet la rencontre du symbole « Ans » (message envoyé par Alpro à l'écran), élément déclencheur du processus.
Mais l'usage de la mémoire Ans~~Sto~~ peut aussi s'opposer à ce processus comme nous l'avons observé chez le premier binôme. En effet, l'appui sur la touche « = » pour avoir le résultat d'un sous calcul, efface le nombre stocké intentionnellement dans le mémoire Ans.
Ces études permettent de mettre en évidence trois 3 usages de la mémoire « Ans » :

- En tant que mémoire Ans~~Sto~~ : le symbole « Ans » est visible dans l'expression sans que la touche « Ans » soit tapée ;
- En tant que mémoire du dernier résultat : le symbole « Ans » est réutilisée (donc la touche est retapée), une *seule fois* dans un calcul ;
- En tant que mémoire variable mathématique : le symbole « Ans » est réutilisée (donc la touche est retapée), *plusieurs fois* dans un calcul

D'autres touches doivent être instrumentées corrélativement aux touches mémoires, en particulier les touches de correction AC/ON, C/OFF et les touches d'initialisation des mémoires comme les touches « = » pour Ans et Sto pour les mémoires variables.

Les calculs proposés et les difficultés à les programmer avec Alpro, peuvent limiter le processus d'instrumentation aux seules touches AC/ON et C/OFF : par exemple, le premier binôme a appris à les différencier.

L'instrumentation des touches de correction, en particulier de AC/ON, peut surajouter une signification non prévue par les constructeurs à l'usage des mémoires : l'appui sur la touche « = », à la suite de l'entrée d'un nombre v, initialise la mémoire Ans, l'appui sur AC/ON conserve le nombre v, une fois devenu invisible, dans cette mémoire.

Chapitre 2

Première partie
Analyse *a priori* de la situation 2
Programme à une machine (Alpro, CALCULATOR)
proche de la machine analytique de Babbage

Introduction

L'enjeu de cette situation est la formulation de l'invariant opératoire d'un calcul répétitif (*tabulation d'une fonction*) dans un message à une machine Alpro bloquée seule, puis associée à un robot CALCULATOR, le couple (Alpro, CALCULATOR) *ayant un fonctionnement proche de la machine analytique de Babbage*.
Cette formulation d'un calcul répétitif à une machine est une condition nécessaire, d'après nos hypothèses, à la conceptualisation des notions de variable informatique et de boucles.

Rappelons deux hypothèses que nous avons explicitées à la fin de la partie B de notre travail et qui fondent la conception de la situation 2 :
Hypothèse
L'écriture d'un programme de calculs répétitifs (algorithme itératif) en un langage proche du fonctionnement de la machine est une condition favorable à l'émergence de la notion de variable comme mémoire effaçable.
Hypothèse
La formulation de la condition d'arrêt et de l'invariant de la répétition, nécessaire à l'écriture d'un message à une machine à mémoire effaçable, oblige à un travail réflexif sur les objets mathématiques présents dans la solution mathématique d'un problème.

Dans les situations 2 et 3, le problème mathématique de tabulation est le suivant :

> Soit **f la fonction $x \rightarrow f(x) = x^2 + 1$.**
> Calculer les images par cette fonction de m nombres x_0, $x_1 \ldots x_k \ldots$ espacées d'un pas p et appartenant à l'intervalle **[-3, 2] avec $x_0 = -3$**

Nous avons choisi :

- une fonction non problématique et présente dans les chapitres sur les fonctions des manuels à partir de la classe de Seconde dans les deux institutions française et vietnamienne.
- l'intervalle [-3, 2] est tel que les bornes sont des entiers « petits », l'un négatif, l'autre positif en conformité avec la donnée des intervalles de définition d'une fonction dans les deux institutions.
- l'intervalle [-3, 2] est non symétrique par rapport à 0 pour éviter la réduction de la répétition par l'utilisation d'une propriété mathématique : f étant une fonction paire, si [a, b] est symétrique par rapport à 0, le nombre de calculs des images peut être réduit de moitié par symétrie.

Nous avons joué sur *la taille du pas relativement à la longueur de l'intervalle* pour organiser un saut informationnel entre la phase 1 et la phase 2 de la situation 2.

Nous allons maintenant rentrer plus dans le détail des phases de la situation 2.

I. Situation 2 – Phase 1 : un premier problème de tabulation

La phase 1 de la situation 2 a pour objectif de faire la dévolution du problème mathématique de tabulation d'une fonction. L'observable de cette dévolution est le résultat (individuelle) du calcul des élèves.

I.1. Énoncé du premier problème de tabulation (phase 1)

Phase 1 (15 minutes)
Exercice 1 (10 minutes). Travail individuel avec la calculatrice Alpro ou la calculatrice personnelle. Avec la calculatrice personnelle, utiliser les mêmes touches autorisées qu'Alpro
Soit la fonction : $x \rightarrow f(x) = x^2 + 1$.
On veut calculer les images, par la fonction *f*, de plusieurs valeurs de x appartenant à l'intervalle [-3, 2]. Pour cela, on prend un écart[1] fixe entre deux valeurs successives de x, en commençant par -3. Les premiers calculs se présentent ainsi :

f(-3) = 10 ;
f(-2,8) = 8,84 ;
f(-2,6) = 7,76 ;

a) Quel est l'écart choisi entre deux valeurs successives de x ?
b) Calculez les images par la fonction f de la sixième, la onzième et la treizième valeur de x.

I.2. Les choix et les stratégies possibles

Dans cette première phase, le pas p (à trouver) est 0,2.
On demande le calcul de trois valeurs parmi 13 valeurs successives appartenant à cet intervalle. Le nombre maximum de calculs (répétitions) pour cette tabulation est donc de 13.
Les images des trois premières valeurs par la fonction f sont données mais sous une forme non « traditionnelle » (pas de tableau de valeurs), afin de ne privilégier aucun procédé de calcul. En effet, la présence d'un tableau de valeurs peut induire une stratégie liste (liste exhaustive des x_i et des $f(x_i)$) et un mode de calcul de proche en proche.

Le tableur - calculatrice a été volontairement omis dans la conception d'Alpro ou interdit sur la calculatrice personnelle (seules les touches semblables à celles d'Alpro sont permises), *pour bloquer le recours à des tableurs.*
En effet, le tableur prend en charge la répétition et représente la solution institutionnelle à ce type de problème dans les classes de Seconde en France (cf. chapitre A2).

Cette phase est une situation d'action (Brousseau 1998), au sens, où l'on peut répondre à la question sans avoir à expliciter la répétition.

Deux algorithmes implicites (de cette répétition) sont possibles :
- algorithme M(R) correspondant à la formule de récurrence : $x_{k+1} = x_k + p$ avec $x_0 = -3$
- algorithme M(F) correspondant à la formule fonctionnelle : $x_{k+1} = a + k.p$

Deux stratégies de calculs sont associées ces deux algorithmes implicites :
- M(R) : calculer le nombre x_i en ajoutant la valeur p au nombre x_{i-1} (calculé précédemment) en commençant par x = -3 puis calculer $f(x_i)$.

[1] Dans l'énoncé nous avons choisi le mot « écart » au lieu de « pas » pour désigner le nombre p car le mot « pas » n'est pas connu dans l'institution vietnamienne. Dans l'analyse, nous utilisons le mot « pas ».

- M(F) : calculer les nombres $x_{n+1} = a + np$ puis calculer $f(x_n)$.

Des observables de M(R) peuvent être :

- un tableau de valeurs,
- une liste ordonnée de nombres où l'on passe d'un nombre à l'autre en ajoutant successivement p pour obtenir x_6, x_{11} et x_{13}.

Un observable de M(F) est la présence d'un schéma de calcul donnant indépendamment du tableau les valeurs x_6, x_{11}, et x_{13} et interprétable par $x_{k+1} = a + k.p$.
La consigne « Calculez les images par la fonction f de la sixième, la onzième et la treizième valeur de x » et la donnée de la première valeur – 3 peut permettre l'apparition de réponses basées sur l'algorithme fonctionnel. Mais cette apparition peut donner lieu à des erreurs ou à des difficultés puisque l'algorithme s'écrit : $x_{n+1} = - 3 + 0,2n$. Le décalage des indices exige un travail dans l'initialisation des variables lors de l'écriture des programmes dans le langage machine Alpro.

On peut tenter le calcul direct des images (c'est-à-dire sans passer par le calcul des x_i) selon les deux algorithmes. Le choix de la fonction f rendant « complexe » ce calcul, on peut prévoir des erreurs du type linéaire :
Par exemple : $f(x + 0,2) = f(x) + 0,2$ au lieu de $f(x + 0,2) = f(x) + 0,4x + 0,04$

À la fin de ce premier calcul répétitif, l'enseignant effectue une synthèse dans laquelle il se contente d'écrire les réponses correctes au tableau sans donner d'explications.

II. Situation 2 – Phase 2 : écriture d'un programme de calcul répétitif à une machine

L'enjeu de cette phase est d'écrire un programme de calcul répétitif à une machine et donc de rencontrer, à propos d'un problème de tabulation, un problème informatique. Le langage pour l'écriture de ce programme est le *langage Alpro*[1] (déjà initié lors du calcul 3 de la situation 1).

II.1. Énoncés du deuxième problème de tabulation et du problème informatique (phase 2)

Phase 2 (15 minutes)
Travail en binôme avec papier, stylo et Alpro (10 minutes)

On veut maintenant réduire l'écart entre deux valeurs successives de x en le prenant égal à 0,03. On commence les calculs avec la calculatrice Alpro :

f(-3) = 10 ;

f(-2,97) = 9,8209 ;

Il y a beaucoup de calculs à répéter !

On fait alors appel au robot CALCULATOR qui sait faire deux actions :
• Appuyer sur les touches de la calculatrice Alpro à condition qu'on lui indique ces touches par écrit ;
• Imprimer automatiquement ce qui est affiché sur la ligne de résultat de la calculatrice Alpro après l'appui sur la touche [=] ;

CALCULATOR ne sait faire que ces deux actions et ne possède ni papier ni stylo

On décide d'écrire un message au robot CALCULATOR en lui indiquant les touches de la calculatrice Alpro sur lesquelles appuyer.
CALCULATOR, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2[et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel f : $x \rightarrow f(x) = x^2 + 1$

Écrivez ce message :

[1] Suite de touches d'Alpro.

II.2. Le saut informationnel organisé entre la phase 1 et la phase 2

La valeur p = 0,03 a été choisie de telle sorte que :

- Le résultat de la division $\frac{b-a}{p}$ ne soit pas un nombre entier (166,66…) : aucun x_k ne prendra la valeur 3. La formulation d'une condition d'arrêt devient plus problématique et oblige à un retour réflexif sur des notions mathématiques comme les notions d'intervalle, de fonction et d'image d'un nombre par une fonction.
- Le nombre des x_i soit grand (167), et donc, de la même façon, le nombre de calculs à répéter pour donner les images de ces nombres. Par exemple, en suivant l'algorithme implicite de la récurrence, il faut $167 \times 2 = 334$ additions et 167 multiplications.
- Le nombre d'appuis de touches à écrire dans le message à la machine, qui est aussi la longueur du message, est très grand (compris entre 1800 et 3600 appuis). L'écriture d'un tel message sans structure de répétition devient très coûteuse, voire impossible.

Le passage de la valeur 0,2 à la valeur 0,03, augmente donc drastiquement, non seulement la complexité du calcul (saut informationnel) mais aussi l'écriture du programme.

Le binôme récepteur du programme en langage Alpro du calcul 3 (situation 1) a été remplacé par un robot fictif CALCULATOR.
Ce robot a pour seules capacités :

- d'exécuter un calcul donné en langage Alpro ;
- d'imprimer automatiquement un résultat après l'appui sur la touche [=] d'Alpro.

Ainsi pour chaque calcul d'une image par f d'un nombre x_i, la touche [=] ne peut être appuyée que pour imprimer un résultat.
En attribuant ces deux fonctions à CALCULATOR, la machine (Alpro, CALCULATOR) possède des capacités communes avec celles la machine analytique de Babbage :

- Une unité de calcul : la machine arithmétique Alpro ;
- Une unité de mémoire qui contient des mémoires variables et Ans : la machine à mémoire effaçable Alpro ;
- Une unité de commande fictive CALCULATOR qui permet l'exécution automatique d'un calcul non répétitif ;
- Une sortie impression fictive pour l'écriture des résultats finaux de calcul suite à l'appui de la touche [=] ;
- Une entrée fictive avec lecture d'un programme en langage Alpro : le robot CALCULATOR sait lire un message comportant les touches de la calculatrice Alpro et appuyer sur ces touches dans l'ordre indiqué dans le message. Ceci n'est pas détaillé dans la consigne à l'élève : on fait l'hypothèse qu'il interprétera la consigne dans ce sens.

Mais, contrairement à la machine analytique de Babbage, cette machine ne peut pas prendre en charge la répétition : si on respecte les capacités de CALCULATOR, la longueur du programme (nombre de touches présentes dans le programme) est exactement le nombre d'appuis sur les touches d'Alpro que CALCULATOR doit exécuter.
L'écriture du message est donc impossible.

Cependant on peut envisager l'écriture *du début* du message et l'utilisation de signes (non compris par CALCULATOR) pour indiquer la répétition : « … », « ainsi de suite », « etc. ».

Quelles sont les écritures possibles de l'invariant de la répétition dans le langage Alpro ? C'est ce que nous allons maintenant examiner.

II.3. Les écritures possibles en langage Alpro de l'invariant de la répétition

Rappelons que les mémoires variables A et B d'Alpro ne prennent pleinement le statut de variables informatiques qu'à travers l'affectation successive et intentionnelle de nombres dans ces mémoires. *Dans ce cas, nous les appelons « variable » au lieu de « mémoire variable ».*

Nous donnons, dans le tableau l, les écritures possibles en langage Alpro pour la machine (Alpro, CALCULATOR) de l'invariant de la répétition selon les deux algorithmes M(R), M(F).

	Algorithme M(R)	**Algorithme M(F)**
• Initialisation de A • Invariant de la répétition	- 3 Sto A $\begin{cases} A \times A + 1 = \\ A + 0{,}03 \quad Sto \quad A \end{cases}$	0 Sto A $\begin{cases} -3 + A \times 0{,}03 \quad Sto \quad B \\ B \times B + 1 = \\ A + 1 \quad Sto \quad A \end{cases}$

Tableau 1. Écriture en langage Alpro de deux invariants de la répétition selon M(R) ou M(F)

Dans l'invariant de la répétition pour l'algorithme M(F), le blocage des parenthèses oblige à l'usage de deux mémoires, la mémoire B permettant de stocker le nombre - 3 + A × 0,03.

Ces écritures de l'invariant mettent en place deux opérations fondamentales sur les variables informatiques dans un programme :

- l'initialisation de la variable A, commune aux deux programmes ; l'initialisation de la variable B lors de la première exécution de l'invariant basé sur l'algorithme M(F).
- la mise à jour des variables : par exemple, A + 1 Sto A

Cette mise à jour est un observable du fonctionnement d'une mémoire variable comme variable (informatique).
Ces deux opérations s'appuient sur l'affectation par appui sur la touche « Sto », déjà exposée publiquement dans les synthèses de la situation 1, en particulier lors du calcul 3.

L'appui sur la touche « Sto » pour stocker un nombre met automatiquement ce nombre dans la mémoire Ans : les groupes de touches « A × A + 1 » et « B × B + 1 » peuvent être remplacés par « Ans × Ans + 1 ».
Cependant l'usage de la mémoire Ans comme seule mémoire dans le programme n'est théoriquement pas possible, la répétition du calcul de l'image d'un nombre exigeant la mise à jour d'une mémoire contenant ce nombre qui ne peut pas être Ans.

II.4. Institutionnalisation des capacités et des limites du Robot CALCULATOR

À la fin de cette phase, l'enseignant organise la présentation publique de messages choisis dans le dessein de préciser les capacités et les limites de CALCULATOR :

- *Ce qu'il peut faire* : exécuter une suite de touches et imprimer automatiquement chaque fois que la touche « = » est pressée ;
- *Ce qu'il ne sait pas faire* : comprendre un mot ou un symbole.

L'enseignant conclut « Étant données les capacités de CALCULATOR, la longueur du programme (nombre d'appuis de touches d'Alpro) est égale au nombre d'appuis de touches d'Alpro. On ne peut donc pas écrire de message à la machine (Alpro, CALCULATOR) dans le temps dont on dispose. »

III. Situation 2 – Phase 3 : coût d'un programme écrit en langage Alpro à la machine (Alpro, CALCULATOR)

Cette phase a pour objectifs de permettre :

- l'identification de l'invariant de la répétition du calcul dans le deuxième problème de tabulation. L'élève n'a pas à le formuler, ni l'enseignant à l'institutionnaliser.
- l'émergence en actes d'éléments sur la complexité des programmes utilisés : nombre d'opérations, nombre de mémoires. Ce travail sur le coût d'un message prépare la situation suivante qui vise la formulation d'une condition d'arrêt d'une itération et, pour se faire, un retour réflexif sur les notions mathématiques, comme fonction et intervalle.

III.1. Énoncé de la phase 3

Phase 3 (15 minutes) Travail en binôme avec papier, stylo et Alpro

La longueur du message à CALCULATOR est égale au nombre d'appuis de CALCULATOR sur Alpro.
Combien de fois en tout CALCULATOR appuie-t-il sur les touches de la calculatrice Alpro ?

Nombre de fois : ☐

Expliquez comment vous avez calculé ce nombre :

III.2. Coût d'un programme écrit en langage Alpro à la machine (Alpro, CALCULATOR)

III.2.1. Dénombrement des appuis de touches

Ce dénombrement demande un travail réflexif sur la notion d'intervalle et la notion de fonction.

En effet, il demande

- d'identifier un invariant de la répétition du calcul de la tabulation ;
- puis de dénombrer le nombre de répétitions de cet invariant, ce qui revient à dénombrer soit les nombres x_i appartenant à [-3 ; 2], soit les nombres $f(x_i)$ appartenant à l'intervalle image.

Le dénombrement de nombres appartenant à un intervalle réel n'est pas habituel au lycée, le tableau de valeurs, et *a fortiori* celui des tableurs, prenant institutionnellement en charge ce travail. Dans ce dénombrement, un intervalle n'est pas réduit à ses bornes, plus on diminue le pas, plus il contient plus de x_i (passage du pas 0,2 au pas 0,03).

Si on considère l'intervalle [-3 ; 2], le dernier x_i n'est pas égal à 2, ce qui est une rupture du contrat didactique par rapport aux pratiques de subdivision d'un intervalle.

L'intervalle image [1, 10] demande de chercher les valeurs minimum et maximum de f sur l'intervalle [-3 ; 2], le maximum n'étant pas l'image de 2. Le décompte n'est pas possible sur les $f(x_i)$, puisque les x_i de l'intervalle [-3 ; 2] symétriques par rapport à 0 ont la même image.

Dénombrer le nombre de répétitions de l'invariant de calcul revient à dénombrer les x_i appartenant à l'intervalle [-3 ; 2] en commençant par -3.

Ce nombre est $n = E\left(\frac{2-(-3)}{0,03}\right)+1$ où E(x) est la partie entière d'un réel x.

Ici n = E(166,666…) + 1 = 167.

III.2.2. Coût selon l'algorithme de calcul

La consigne « imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et *seulement* ces images » doit être interprétée comme « l'appui sur la touche « = » est autorisé seulement une fois pour imprimer $f(x_i)$ ».

Le calcul du coût des programmes implique le calcul du nombre d'appuis de touches.

En nous référant aux deux algorithmes M(R) et M(F) et à l'écriture de l'invariant de la répétition en langage Alpro (tableau 1), nous pouvons calculer le coût de ces deux algorithmes.
Pour calculer les images par *f* de n nombres x_i :

- l'algorithme M(R) nécessite 2n additions, n multiplications et une variable donc une place dans la mémoire
- l'algorithme M(F) exige 3n additions, 2n multiplications et 2 variables, c'est-à-dire 2 places dans la mémoire.

L'algorithme M(R) est donc moins coûteux que l'algorithme M(F).

Il existe des variantes moins coûteuses, obtenues par utilisation d'une mémoire supplémentaire contenant la valeur du pas: cette mémoire supplémentaire est une mémoire mathématique (son contenu ne change pas). Ceci permet de diminuer le coût du programme du coût de l'entrée du nombre 0,03 (4 appuis – 1 appui pour la mémoire du pas) : 3 × 167 = 501.

Nous résumons dans le tableau 2 ci-après les nombres d'appuis de touches possibles selon les algorithmes M(R) ou M(F) et nous donnons les variantes.

	Algorithme M(R)	**Algorithme M(F)**
• Initialisation de A • Invariant de la répétition	- 3 Sto A $\begin{cases} A \times A + 1 = \\ A + 0{,}03 \ \ Sto \ \ A \end{cases}$	0 Sto A $\begin{cases} -3 + A \times 0{,}03 \ \ Sto \ \ B \\ B \times B + 1 = \\ A + 1 \ \ Sto \ \ A \end{cases}$
Nombre d'appuis de touches	4 + 14 × 167 = 2342	3 + 22 × 167 = 3677
Amélioration du coût par utilisation d'une mémoire variable mathématique supplémentaire	-3 Sto A 0,03 Sto B $\begin{cases} A \times A + 1 = \\ A + B \ Sto \ A \end{cases}$	-3 Sto A 0,03 Sto C $\begin{cases} -3 + A \times C \ Sto \ B \\ B \times B + 1 = \\ A + 1 \ Sto \ A \end{cases}$
Nombre d'appuis de touches	4 + 6 + 11× 167 = 1847	3 + 6 + 19 × 167 = 3182

Tableau 2. Nombres d'appuis de touches suivant les deux algorithmes M(R) et M(F)

Remarquons qu'écrire « A × A + 1 » ou « Ans × Ans+ 1 », ne change pas le nombre d'appuis de touches.

La consigne « imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et *seulement* ces images » peut être interprétée comme « l'appui sur la touche « = » permet d'imprimer un résultat ». Ce qui revient à :

- soit, imprimer les nombres x_i et leurs images par f (interprétation 1).
- soit, imprimer les nombres x_i (interprétation 2, conséquence de l'usage exclusif de la mémoire Ans)

Dans ce cas, on peut avoir deux autres invariants de la répétition :

Interprétation 1 algorithme M(F)	Interprétation 2 algorithme M(R)
$\begin{cases} -3 + A \times 0{,}03 = \\ Ans \times Ans = \\ A + 1 \ Sto \ A \end{cases}$	- 3 = + 0,03 = {=

Tableau 3. Deux invariants de la répétition selon deux interprétations

Pour les programmes suivant l'interprétation 1, les coûts précédents diminuent de 167 appuis sur la touche « = », touche « = » qui remplace les deux touches « Sto » et « B ».
Pour les programmes suivant l'interprétation 2, le nombre d'appuis de touches devient : 9 + 165 = 174.
Lors qu'on initialise la mémoire Ans par « - 3 = », puis tape « + 0,03 », sur la ligne d'expression d'Alpro apparaît « Ans + 0,03 » et sur la ligne d'état « - 2,97 ». Puis on appuie successivement sur « = » pour obtenir les nombres : - 2,94 ; - 2,91 etc.

IV. Situation 2 – Phase 4 : amélioration du Robot CALCULATOR ?

Cette phase initie le processus de conception d'un nouveau Robot, CALCULATOR II, qui permette à la machine (Alpro, CALCULATOR) de prendre en charge la répétition d'un problème de tabulation et qui sera l'enjeu de la situation 3.
Comment faire pour que la longueur du message soit très inférieure au nombre d'appuis de CALCULATOR sur les touches Alpro ? Comment faire pour rendre possible, dans un temps limité, l'écriture d'un message à la machine ?

IV.1. Énoncé de la phase 4

Phase 4 (15 minutes)
Travail en binôme avec papier, stylo et Alpro (10 minutes)
On va essayer dans la séance suivante d'améliorer le robot CALCULATOR pour rendre la longueur du message inférieure au nombre d'appuis de CALCULATOR sur Alpro. Cela nous permettra non seulement d'écrire tout le message mais aussi de le rendre considérablement plus court.
Pour améliorer le robot CALCULATOR, que proposez-vous ?
Écrire vos propositions :

IV.2. Améliorations possibles du Robot CALCULATOR

On peut envisager deux grands types d'amélioration du Robot CALCULATOR :

- Amélioration 1 : pouvoir répéter l'exécution d'un groupe de touches d'Alpro et donc *comprendre l'ordre de répétition ;*
- Amélioration 2 : même capacité que précédemment et en plus comprendre des expressions logiques

À ces deux grands types correspondent des types de boucles possibles.

Cas de l'amélioration 1 : le robot CALCULATOR comprend l'ordre de répétition

Algorithme récurrent M(R)	Algorithme fonctionnel M(F)
Début - 3 Sto A Répéter A × A + 1 = A + 0,03 Sto A Fin répéter Fin	Début 0 Sto A Répéter - 3 + 0,03 × A Sto B B × B + 1 = A + 1 Sto A Fin répéter Fin

Tableau 3. Exemple de boucle si CALCULATOR comprend l'ordre de répétition

Cas de l'amélioration 2 : le robot CALCULATOR comprend l'ordre de répétition et les expressions logiques

Algorithme récurrent M(R)		
Boucle « tant que »	**Boucle « retourner à »**	**Boucle « répéter »**
Début - 3 Sto A Tant que A ≤ 2 A × A + 1 = A + 0,03 Sto A Fin tant que Fin	10. Début 20. - 3 Sto A 30. A × A + 1 = 40. A + 0,03 Sto A 60. Si A ≤ 2 retourner à 30 70. Fin	Début - 3 Sto A Répéter A × A + 1 = A + 0,03 Sto A Jusqu'à A > 2 Fin

Tableau 4. Exemple de boucle si CALCULATOR comprend l'ordre de répétition et les expressions logiques [M(R)]

Algorithme fonctionnel M(F)		
Boucle « tant que »	**Boucle « retourner à »**	**Boucle « répéter »**
Début 0 Sto A Tant que – 3 + 0,03 × A ≤ 2 – 3 + 0,03 × A Sto B B × B + 1 = A + 1 Sto A Fin tant que Fin	10. Début 20. 0 Sto A 30. – 3 + 0,03 × A Sto B 40. B × B + 1 = 50. A + 1 Sto A 60. Si B ≤ 2 retourner à 30 70. Fin	Début 0 Sto A Répéter – 3 + 0,03 × A Sto B B× B + 1 = A + 1 Sto A Jusqu'à B > 2 Fin

Tableau 5. Exemple de boucle si CALCULATOR comprend l'ordre de répétition et les expressions logiques [M(F)]

Nous prévoyons que le calcul du nombre d'appuis de touches de la phase 3 favorise la première amélioration, qui est aussi celle l'amélioration « la plus simple » du Robot.

Une synthèse est prévue à la fin de cette phase durant laquelle l'enseignant passe en revue des propositions faites par des binômes afin de conclure que le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution d'un groupe de touches d'Alpro.

V. Conclusion de l'analyse *a priori* de la situation 2

Pour conclure, nous nous contentons de schématiser les quatre phases de la situation 2 dans la figure 1.

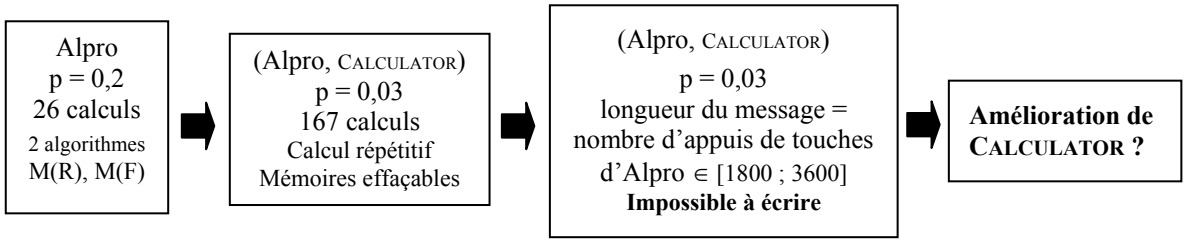


Figure 1. Les quatre phases de la situation 2 de l'ingénierie didactique

Que s'est-il passé ?
Nous allons maintenant analyser les réalisations de la situation 2 dans EMS en France et au Viêt-nam pour répondre à cette question.

Chapitre 2

Deuxième partie
Analyse *a posteriori* de la situation 2

Comme déjà précisé dans la présentation des conditions et du recueil des données de l'expérimentation (Chapitre D1, Deuxième partie) l'ensemble de l'ingénierie didactique a été réalisé dans les deux classes de 1er F et de 1er V (respectivement 18 élèves et 20 élèves).

Les élèves travaillent en binômes sur Alpro dans toutes les phases sauf durant la phase 1 de la situation 2 où les élèves de 1er V travaillent individuellement (pour moitié avec la calculatrice Alpro, pour moitié avec la calculatrice personnelle).

I. Phase 1 : un premier problème de tabulation

Nous rappelons l'énoncé de cette phase.

Phase 1 (15 minutes)
Exercice 1 (10 minutes). Travail individuel avec la calculatrice Alpro ou la calculatrice personnelle. Avec la calculatrice personnelle, utiliser les mêmes touches autorisées qu'Alpro
Soit la fonction : $x \rightarrow f(x) = x^2 + 1$.
On veut calculer les images, par la fonction *f*, de plusieurs valeurs de x appartenant à l'intervalle [-3, 2]. Pour cela, on prend un écart fixe entre deux valeurs successives de x, en commençant par -3. Les premiers calculs se présentent ainsi :

f(-3) = 10 ;
f(-2,8) = 8,84 ;
f(-2,6) = 7,76 ;

a) Quel est l'écart choisi entre deux valeurs successives de x ?
b) Calculez les images par la fonction f de la sixième, la onzième et la treizième valeur de x.

Les résultats attendus sont :

a) l'écart p = 0,2 ;
b) les images par la fonction f de la sixième, de la onzième et de la treizième valeur de x sont respectivement : f(- 2) = 5 ; f(- 1) = 2 ; f(- 0,6) = 1,36.

Concernant la question a), le calcul du pas p n'est pas problématique pour les élèves de ce niveau, et en effet tous les élèves ont donné le résultats correct : p = 0,2.
Concernant la question b), rappelons que deux algorithmes implicites fondent les calculs des images par la fonction f : M(R) $x_{k+1} = x_k + p$ et M(F) $x_{k+1} = a + k \times p$.
Les observés de ces algorithmes sont les fichiers historiques (tous les binômes de 1er F car ils travaillent tous sur Alpro et la moitié des élèves de 1er V), les calculs ou réponses écrites dans les fiches ou les brouillons.

- deux élèves de 1er F n'ont pas donné de résultat écrit ;
- tous les élèves en 1er V ont donné le résultat écrit mais la reconstitution des algorithmes de calcul n'est possible que pour 12 élèves (fiches de réponse, fichiers historiques, brouillons). Huit élèves utilisent la notation des suites qu'on leur a enseignée[1].

Le tableau 1 ci-après donne la répartition des réponses suivant l'algorithme sous-jacent:

[1] Alors que la notion de suite n'a pas encore été enseignée en 1er F

Algorithmes / Classes	Algorithmes M(R)	Algorithmes M(F)	Pas de réponse	Total
1er F	3	7	0	10
1er V	0	12	6	18
Total	3	19	6	28

Tableau 1. Répartition des élèves répartition des réponses suivant l'algorithme sous-jacent

L'algorithme M(F) utilisant le rang i pour calculer la valeur x_i dans la formule fonctionnelle semble implicitement plus disponible chez des élèves que l'algorithme M(R) utilisant un calcul de proche en proche.
L'algorithme M(F) permet de donner directement le terme x_i alors que l'algorithme M(R) oblige au calcul de tous les termes précédant x_i. Or cela exige l'écriture des *13* premières valeurs de x.
Les trois binômes de 1er F calculant suivant l'algorithme M(R) donnent tous des résultats corrects.
Huit résultats (sur 26 : 2 en 1er F et 6 en 1er V) ne sont pas corrects et relèvent de l'algorithme M(F). L'erreur (prévue) réside dans le décalage des indices $x_{n+1} = - 3 + 0{,}2n$.
Par exemple, un élève calcule l'image par la fonction f de la 6e valeur de x comme :

$x = - 3 + 6 \times 0{,}2 = - 1{,}8$ et puis $f(x) = (- 1{,}8)^2 + 1 = 4{,}24$.

Cette erreur est plus fréquente dans l'institution vietnamienne, du fait que le premier terme d'une suite est habituellement noté x_1 et donc $x_k = a + k \times 0{,}2$.
Au contraire dans l'institution française le 1er terme est le plus souvent noté x_0 et donc $x_k = a + (k - 1) \times 0{,}2$, en accord avec ce premier problème de tabulation.

La prédominance de l'algorithme M(F) se maintient-elle dans l'écriture d'un programme de calcul répétitif à la machine (Alpro, CALCULATOR) ?

II. Phase 2 : écriture d'un programme de calcul répétitif à une machine

Nous rappelons l'énoncé de cette phase.

Phase 2 (15 minutes)
Travail en binôme avec papier, stylo et Alpro (10 minutes)

On veut maintenant réduire l'écart entre deux valeurs successives de x en le prenant égal à 0,03. On commence les calculs avec la calculatrice Alpro :

f(-3) = 10 ;

f(-2,97) = 9,8209 ;

Il y a beaucoup de calculs à répéter !

On fait alors appel au robot CALCULATOR qui sait faire deux actions :
• Appuyer sur les touches de la calculatrice Alpro à condition qu'on lui indique ces touches par écrit ;
• Imprimer automatiquement ce qui est affiché sur la ligne de résultat de la calculatrice Alpro après l'appui sur la touche [=];

CALCULATOR ne sait faire que ces deux actions et ne possède ni papier ni stylo

On décide d'écrire un message au robot CALCULATOR en lui indiquant les touches de la calculatrice Alpro sur lesquelles appuyer.
CALCULATOR, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel f : $x \rightarrow f(x) = x^2 + 1$

Écrivez ce message :

II.1. Les programmes écrits

Dix sept binômes (sur 19) ont écrit un programme « impossible » : CALCULATOR ne comprend pas la répétition !

L'élève considère donc le robot CALCULATOR comme un autre élève : il attribue au robot CALCULATOR les compétences et les connaissances qu'il possède lui-même. Ce résultat rejoint des résultats déjà repérés dans les études précédentes de Méjias (1985), Rogalski (1984) chez des élèves dans la période d'alphabétisation de la programmation (op. cité. chapitre A1).

Comme nous nous y attendions, la majorité des programmes (16 sur 19) dans cette phase contiennent le début d'un programme sous forme d'un groupe de touches d'Alpro (langage Alpro) qui est implicitement l'invariant de la répétition.

La répétition, présente dans sept programmes (2 en 1er F et 5 en 1er V), qui utilisent des signes non compris par CALCULATOR signifiant la poursuite du calcul :
- quatre avec le signe « … » (1 en 1er F et 3 en 1er V) ;
- deux (en 1er V) avec des mots « refaire », « répéter » ;
- un (en 1er F) désigne un nombre quelconque par m (voir plus loin) ;

Tous les binômes de 1er F donnent des groupes de touches comme résultats.
Parmi les sept programmes (sur 9) ayant répondu à la question, un seul (1er V) est écrit en langage mathématique suivant l'algorithme M(R):

$f(x_2) = (x_1 - 0.03)^2 + 1$; Avec : $x_1 > x_2$; $x_2 = x_1 - 0.03$.

Dans l'analyse *a priori*, nous avons donné les écritures possibles en langage Alpro de l'invariant de la répétition suivant les deux algorithmes M(R) et M(F), en supposant qu'il pouvait y avoir, en plus de l'interprétation attendue de l'énoncé, deux autres interprétations :
- imprimer les nombres $f(x_i)$ (interprétation attendue) ;
- ou imprimer les nombres xi et leurs images par f (interprétation 1) ;

Ces deux premières interprétations s'appuient sur l'usage des mémoires variables et/ou Ans.
- imprimer les nombres x_i (interprétation 2), conséquence de l'usage exclusif de la mémoire Ans).

	Algorithme M(R)	**Algorithme M(F)**
• Initialisation de A • Invariant de la répétition	- 3 Sto A $\begin{cases} A \times A + 1 = \\ A + 0{,}03 \;\; Sto \;\; A \end{cases}$	0 Sto A $\begin{cases} -3 + A \times 0{,}03 \;\; Sto \;\; B \\ B \times B + 1 = \\ A + 1 \;\; Sto \;\; A \end{cases}$
• Initialisation • Invariant de la répétition	**Interprétation 2 : M(R)'** - 3 = + 0,03 = {=	**Interprétation 1 : M(F)'** 0 Sto A $\begin{cases} -3 + A \times 0{,}03 = \\ Ans \times Ans = \\ A + 1 \;\; Sto \;\; A \end{cases}$

Le tableau 2 ci-après donne la répartition des messages suivant les algorithmes.

Classes \ Algorithmes	Algorithmes M(R)	Algorithmes M(R)'	Algorithmes M(F)	Algorithmes M(F)'	Formule	Non réponse	Total
1er F	4	2	2	2	0	0	10
1er V	3	2	1	0	1	2	9
Total	7	4	3	2	1	2	19

Tableau 2. Répartition des binômes selon les deux algorithmes

Rappelons que les 16 programmes associés aux deux algorithmes utilisent nécessairement les mémoires variables ou Ans.

Le tableau 2 permet d'émettre des résultats suivants.

Écriture en langage Alpro de l'invariant de la répétition, élément d'un programme informatique

L'articulation d'un problème de tabulation d'une fonction connue (p petit par rapport à longueur de l'intervalle [a ; b]) et de l'écriture d'un message à la machine Alpro a conduit 16 binômes (sur 19) à écrire des éléments de programmes comportant le début d'un programme qui peut être identifiable à l'invariant de la répétition.

L'algorithme M(F) perd sa prédominance au profit de l'algorithme M(R)

Douze binômes au moins (sur 19) on abandonné un calcul propre à l'algorithme M(F) pour un calcul M(R) dans le passage de la phase 1 à la phase 2. Ce changement est particulièrement frappant en 1er V où au moins neuf élèves sont passés de l'algorithme M(F) à l'algorithme M(R). Onze programmes en langage Alpro utilisent un calcul M(R), contre cinq pour l'algorithme M(F).
Une raison (voir analyse *a priori*) peut être le plus grand coût de l'algorithme M(F) en termes d'opérations et de variables informatiques écrites, par rapport à l'algorithme M(R).

Résistance de la mémoire Ans

Quatre programmes utilisent exclusivement la mémoire Ans (algorithme M(R)') :

- Deux autres programmes sont écrits par des élèves (1er V) qui ont utilisé exclusivement la mémoire Ans (M_a) dans le calcul 3.
- deux programmes sont écrits par des élèves (1er F) qui ont utilisé les mémoires variables et/ou Ans dans le calcul 3 (stratégies M_{av} et M_{mv}).

Ces quatre programmes M(R)' s'appuient sur une certaine maîtrise de cette mémoire liée au processus d'instrumentation commencé dans la situation 1

Forte présence des mémoires variables

Nous rappelons dans le tableau 3 comment les mémoires ont été utilisées dans les programmes écrits en langage Alpro pour le calcul 3 (situation 1) dans les deux classes.

Programmes / Classes	M_a	M_{av}	M_{vm}	M_v	Pas de réponse	Total
1er F	0	8	1	1	0	10
1er V	5	3	0	0	1	9

Tableau 3. Les programmes écrits en langage Alpro selon l'usage des mémoires dans le calcul 3 (Situation 1)

Pour l'écriture d'éléments de programmes dans le calcul 3 (Situation 1), treize programmes (sur 18 : 10 en 1er F ; 3 en 1er V) utilisaient des mémoires variables A, B, C avec ou sans Ans.
Dans la phase 2 de la situation 2, dans le deuxième problème de tabulation (calcul répétitif), douze programmes (sur 16 : 8 en en 1er F ; 4 en 1er V) utilisent A, B, C en relation avec les algorithmes M(R), M(F) et M(F)'. Il semble donc que cet usage se stabilise.

L'analyse de la situation 1 a mis en évidence la prédominance du statut des mémoires A, B, C comme variables mathématiques : une mémoire variable contient un et un seul nombre durant tout le calcul et peut être utilisé pour écrire algébriquement un calcul à Alpro.

Quel statut ont les mémoires variables présentes dans les programmes de la phase 2 (situation 2) ? Ont-elles conservé le statut des variables mathématiques ou les conditions du nouveau calcul (deuxième problème de tabulation) ont-elles fait évoluer ce statut vers celui de variable informatique ?

Les mémoires variables évoluent vers le statut des variables informatique pour six programmes sur douze.

Pour le montrer, analysons de plus près les douze programmes utilisant des mémoires variables A, B, C (avec ou sans la mémoire Ans).

Forte présence de l'opération d'initialisation d'une variable

L'une des opérations fondamentales concernant la variable informatique est son initialisation. Elle est présente dans les douze programmes utilisant les mémoires variables.
Or, des études précédentes ont montré que « *l'initialisation des variables est une opération cognitive difficile* » ou l'opération d'« *initialisation est globalement celle qui pose le plus de problèmes aux élèves* » (Samurçay 1985, op. cité. chapitre A1). La maîtrise de cette opération chez les élèves observés pourrait s'expliquer par le processus d'instrumentation de la mémoire Ans, et dans une moindre mesure, des mémoires variables, initié dans les calculs 2 et 3 de la situation 1.
Nous donnons ci-après deux exemples d'initialisation.

Algorithme M(F)	**Algorithme M(R)**
- 3 Sto A ; 0,03 Sto B A + B = Ans × Ans +1 A + 2B = Ans × Ans +1 (1^{er} F)	0.03 Sto A ~~- 3 + A = Sto B~~ - 3 + A × A + 1 = AC A + 0.03 = B B × B + 1 = AC … (1^{er} V)

Tableau 4. Deux exemples d'initialisation des variables

Dans le premier programme, utilisant l'algorithme M(F) les deux variables A et B sont initialisées :

- pour A par une valeur « variable » – 3
- pour B par une valeur « constante » q 0,03.

Dans le deuxième programme, la variable B, bien que non nécessaire, est initialisée à l'aide de la variable A.
Mais l'opération d'initialisation ne garantit pas à une mémoire variable le statut de variable informatique.

Une opération fondamentale difficile : la mise à jour

La mise à jour est une opération problématique.

Opérations / Classes	Mise à jour	
	Présente	Absente
1^{er} F	4	4
1^{er} V	2	2

Tableau 5. Répartition des programmes selon l'opération « mise à jour » d'une variable

Ce tableau montre que seulement six programmes mettent à jour les mémoires variables. Or cette opération est liée intrinsèquement à la caractéristique d'effaçabilité des mémoires et donc au statut de mémoire variable.
La restriction du nombre des mémoires à 3 a bien été une condition favorable à l'émergence de la mise à jour d'une mémoire et donc à celle de variable informatique pour ces 6 binômes comme le montrent les deux exemples suivants :

2.97 Sto A A – 0.03 Sto B B – 0.03 Sto C	- 3 + 0.03 Sto A A × A + 1 = AC A + 0.03 Sto B

A × A + 1 = B × B + 1 = C × C + 1 = C – 0.03 Sto A A – 0.03 Sto B B – 0.03 Sto C (1er F)	B × B + 1 = AC B + 0.03 Sto C C × C + 1 = AC C + 0.03 Sto A A × A + 1 = … (1er V)

Les binômes utilisent d'abord toutes les mémoires variables disponibles (A, B et C) avant de se saisir du caractère d'effaçabilité de ces mémoires : elles sont mises à jour, c'est-à-dire leur contenu est changé, seulement lorsqu'il n'y a plus de mémoires variables disponibles.

Absence de la variable « compteur »

L'analyse *a priori* a souligné la complexité de l'algorithme M(F) par rapport à l'algorithme M(R). En effet, l'algorithme M(F) exige la mise en place d'une variable qui n'est pas explicite dans l'énoncé : la variable compteur pour changer le rang i.
Aucun programme utilisant l'algorithme M(F) (5 programmes) n'a explicité la variable compteur et donc la mise à jour de cette variable.
Les deux programmes suivants illustrent cette non prise en compte de la variable « compteur » :

- 3 Sto A ; 0,03 Sto B A + B = Ans × Ans +1 A + 2B = Ans × Ans +1… (1er F)	- 3 Sto A A + 0.03 Sto B B × B + 1 A + 0.06 Sto B B × B + 1 A + 0.09 Sto B B × B + 1… (1er F)

On peut noter que le signe « … » est utilisé dans les deux programmes pour marquer la poursuite du programme.
La caractéristique d'effaçabilité, donc la mise à jour, est prise en compte dans le 2e programme pour la variable B alors qu'elle est absente dans le 1er.

Ce résultat rejoint plusieurs résultats d'études précédentes (Rogalski 1985, Samurçay 1985) :

- une variable absente des données de l'énoncé est difficile à expliciter ;
- la structure « pour i = 1 à n faire » dans laquelle la variation « compteur » i ainsi que son texte ne sont pas explicites, va fonctionner d'abord comme une instruction « miracle » : l'élève se décharge pour la gestion de cette variable sur l'initiative du robot CALCULATOR, assimilé à un humain.

Les programmes en actes et écrits des calculs 2 et 3 et les deux exemples précédents permettent d'identifier une autre raison : la mise à jour de la variable « compteur » est prise en charge par la mémoire papier : « 1, 2… » (1e programme) ou « 0,03 ; 0,06 ; 0,09... » (2e programme).

Transformation difficile de l'invariant de la répétition en un corps de boucle

Malgré la présence des algorithmes M(R) et M(F) dans les 12 programmes, la transformation des invariants en corps de boucle n'est pas présente. Autrement dit, la gestion des variables A, B et C à l'intérieur d'une boucle ne va pas de pair avec l'intention d'initialiser et de mettre à jour ces variables.
Un programme seulement (1er V), utilisant l'algorithme M(R), transforme l'invariant de la répétition en un corps de boucle :

Algorithme M(R)
- 3 Sto → A AC/On A × A + 1 = AC/On A + 0.03 Sto → A AC/On A × A + 1 = AC/On Répéter 164 fois (1er V)

Dans ce programme (1er V), apparaît la condition d'arrêt « Répéter 164 fois ». C'est le seul programme qui le fait.

En résumé

La majorité des binômes ont écrit un programme « impossible » au robot CALCULATOR, celui-ci ne comprenant aucun ordre en dehors de celui de taper sur une touche d'Alpro : il ne peut ni répéter, ni s'arrêter.
Comme nous l'avions prévu, la plupart des binômes écrivent le début du programme qui peut être identifiable à un invariant du calcul répétitif en utilisant des signes incompréhensibles pour le robot CALCULATOR comme « … », soit des mots exprimant la répétition « refaire », « répéter », soit des lettres « m », « n » pour marque la continuation du message.
L'initialisation des mémoires est présente dans tous les programmes utilisant des mémoires variables. Ceci a été préparé par le processus d'instrumentation des touches mémoires variables A, B, C et Ans enclenché dans la situation 1.
Mais la mise à jour n'est présente que dans moins de la moitié des programmes « impossible » : la notion de mémoire effaçable, et donc de variable informatique n'est pas encore présente pour plus de la moitié des élèves.

II.2. Synthèse de l'enseignant sur les programmes proposés

L'enseignant[1] réserve 5 minutes pour faire une synthèse sur les programmes écrits par les binômes.

Dans la classe 1er V

L'enseignant a recopié (entre deux séances) tous les programmes écrits sur 2 feuilles qu'il a distribué aux binômes. Voici la synthèse faite dans cette classe.

> P : […] Je vous distribue alors à chaque binôme deux feuilles sur lesquelles les programmes sont recopiés. Les programmes de 1 à 4 sont dans la 1ère feuille et ceux qui restent dans la 2e feuille. (Il fait circuler les feuilles).
> Es : Les feuilles sont-elles différentes ?
> P : Oui, oui. 2 feuilles pour chaque binôme.

Les huit programmes écrits dans la phase 2 et recopiés par l'enseignant :

[1] Alain Birebent pour la classe 1er F et Nguyen Chi Thanh pour la classe 1er V.

Programme 1 : Appuyer 0.03 Shift, Sto → A AC/On. Appuyer – 2.97 + A appuyer = Ans × Ans + 1 = 9.6434 Les étapes suivantes sont pareilles. Appuyer AC/On puis recommence. **Programme 2 :** $f(x_2) = (x_1 - 0.03)^2 + 1$ avec : $x_1 > x_2$. $x_2 = x_1 - 0.03$. **Programme 3 :** 0.03 → Sto → A $x_1 = -3 \rightarrow x_2 = -3 + A \rightarrow x_3 = -3 + 2A \rightarrow \ldots x_n = -3 + (n - 1)A$ $f(x_n) = [(n - 1)A - 3]^2 + 1$ $= (n - 1)^2A^2 - 6(n - 1)A + 10$ **Programme 4 :** - 3 + 0.03 Sto A A × A + 1 = AC A + 0.03 Sto B B × B + 1 = AC B + 0.03 Sto C C × C + 1 = AC C + 0.03 Sto A A × A + 1 = …	**Programme 5 :** ~~Sto~~ ~~0.03 Sto A~~ ~~⇔ 3 + A = Ans~~ **Programme 6 :** 0.03 Sto A ~~- 3 + A = Sto B~~ - 3 + A × A + 1 = AC A + 0.03 = B B × B + 1 = AC … **Programme 7 :** - 3 Sto → A AC/On A × A + 1 = AC/ON A + 0.03 Sto → A. AC/ON A × A + 1 = AC/ON Répéter 164 fois **Programme 8 :** 0.03 = Ans – 3 + Ans

128. P : Est-ce que tous les binômes ont reçu le 2 feuilles ?
129. Es : Oui.
130. P : Vous le voyez, dans la 1ère feuille, ce sont des programmes de 1 à 4 et dans la 2e de 5 à 8. Ce sont des programmes que vous avez rédigés la dernière fois. Je vous rappelle l'énoncé. Vous devez écrire un programme à un robot qui s'appelle CALCULATOR. Il ne sait faire que deux choses : la première chose… (il cherche l'énoncé) Je vous lis la consigne « Appuyer sur les touches d'Alpro à condition que ces touches lui sont indiquées par écrit, imprimer automatiquement ce qui affiché sur la ligne de résultat après avoir appuyé sur la touche [=] » Voilà les deux choses qu'il peut faire. Maintenant, voyons le programme 1 (Il relit le programme et l'écrit au tableau) « Appuyer 0.03 Shift, Sto A. » Est-ce que CALCULATOR comprend cette phrase ?
131. Es : Oui.
132. P : Vous voyez que ceci il ne le comprend pas. Le mot « appuyer » car il peut pas lire ce mot. De même, la virgule « , » il ne comprend pas non plus. Il suffit d'écrire « 0.03 Shift Sto A » car vous savez que CALCULATOR ne peut que lire les touches d'Alpro.

L'enseignant écrit au tableau :

~~Appuyer~~ 0.03 Shift ~~,~~ Sto

133. P : Ou la 2e phrase « les étapes suivantes sont pareilles ». Est-ce qu'il peut le comprendre ?
134. Es : Non.
135. P : C'est la même chose avec la phrase « puis recommence ». Voyons le 2e programme « $f(x_2) = (x_1 - 0.03)^2$ ». Il ne peut pas comprendre non plus les lettres f, les parenthèses, x_1, x_2 et le carré aussi car ça n'existe pas sur Alpro. En brève, jusqu'à maintenant vous ne pouvez écrire que des programmes contenant les touches d'Alpro.

L'enseignant écrit au tableau :

~~Appuyer~~ 0.03 Shift ~~,~~ Sto
$f(x_2) = (x_1 - 0.03)^2$

136. P : En ce qui concerne le programme 4. La 1ère ligne « - 3 + 0.03 Sto A », il peut la comprendre ?
137. Es : Oui.
138. P : C'est dans le programme 4. « A × A + 1 = » ça il peut le faire. De même «AC A + 0.03 Sto B B × B + 1 = AC B + 0.03 Sto C C × C + 1 = » ça peut être compris par Alpro. C'est pareil pour « AC C +

0.03 Sto A ». Mais en ce qui concerne cette ligne « A × A + 1 = … » (il le note au tableau). Ce qu'il ne comprend pas c'est quoi ?

139. Es : Les trois points.
140. P : Oui, c'est ça, ce sont ces trois point (il entoure d'un cercle les 3 points).

L'enseignant écrit au tableau :

4) A × A + 1 = ...

141. P : C'est-à-dire…Qu'est-ce qu'ils veulent dire par là ?
142. E1 : Continuer
143. E2 : Etc.,
144. E3 : Continuer ou faire jusqu'à sortir le résultat.
145. E4 : Répéter 164 fois. 164 fois à répéter.
146. P (à l'élève qui parle de la répétition) : On répète combien de fois ?
147. E : Répéter 164 fois.
148. P (note au tableau) : Donc les trois points c'est-à-dire 164 fois. Mais vous le savez, CALCULATOR avec ses capacités à présent, il ne peut pas encore comprendre ça. Donc on doit tout lui écrire.

L'enseignant écrit au tableau :

164 fois

149. P : Maintenant, passons au programme 7. Intéressons nous à la dernière ligne du programme « Répéter 164 fois » (et il note en même temps cette phrase au tableau). Il peut comprendre ça le mot « répéter » ?

L'enseignant écrit au tableau :

7) Répéter 164 fois

150. Es : Non.
151. P : Non. Comme vous le voyez, CALCULATOR ne sait faire que 2 choses, premièrement appuyer les touches d'Alpro et deuxièmement imprimer. Dans cette phrase, nous avons vu aussi que les capacités d'Alpro, il faut écrire tous les calculs et c'est très long à écrire, beaucoup de répétitions. On va passer à la phase suivante. Je vous distribue donc les nouvelles feuilles. […]

L'enseignant n'a pas exécuté les programmes sur Alpro mais rappelé publiquement les limitations du robot CALCULATOR qui n'a que deux capacités: « appuyer sur des touches d'Alpro » et « imprimer automatiquement à la suite de l'appui sur la touche = ». Tous les programmes écrits ont été examinés du point de vue des deux capacités de CALCULATOR.

Dans la classe de 1er F

L'enseignant a joué le rôle du robot CALCULATOR, en invitant un binôme de venir lui dicter le programme écrit afin de l'exécuter sur la calculatrice Alpro publique.

85. P : C'est bon ? Je peux les ramasser, s'il vous plaît ? C'est bon là. Je ramasse quand même. […] Je vous propose que c'est moi qui joue le CALCULATOR et vous viendrez ici pour me donner des programmes. Voilà je leur redonne le programme et moi je suis CALCULATOR. Je sais parfaitement ce que je sais faire, ah ? Donc tout le monde peut voir ? Je vous écoute et je suis CALCULATOR ah ? J'attends les touches.

Le programme écrit par le binôme invité au tableau est le suivant :

- 3 Sto A
0.03 Sto B
A + B = × Ans + 1
A + B × 2 = × Ans + 1
A + B × 3 = × Ans + 1
A + B × m = × Ans + 1

86. E (au tableau) : – 3 Sto A.
87. P : Oui là, je sais faire. Je pense que c'est le moins là.

88. E : – 3 Sto A AC/ON[1]
89. P : Pour l'instant je sais faire car les touches je sais faire et je comprends. AC/ON.
90. E : Ensuite 0.03 Sto B et AC/ON.
91. P : AC/ON pour l'instant ce sont des touches et moi, je comprends.
92. E : Ensuite A plus la mémoire B et on fait égale.
93. P : Egale. Oui.
94. E : Et sur le résultat on fait AC/ON et on fait Ans fois Ans plus 1.
95. P : OK. Vous avez suivi un peu les opérations là ? Ans fois Ans et plus 1. Je fais quoi là ? Je fais simplement. Ah, voilà, Ans fois Ans plus 1. Egale et puis ?
96. E : Et après je recommence ça on refais et puis…
97. P : Attends, attendez. Moi, je suis CALCULATOR
98. Es (rient) :
99. P : Et je ne comprends que…
100. E : Bah, on fait égale et Stocké.
101. P : Je fais égale là ? Ouis ? Et puis, qu'est-ce que je fais ?
102. E : Non, AC/ON. Et après CALCULATOR il lit le résultat.
103. P : Non, non, je suis CALCULATOR et je sais faire des choses et je l'ai dit.
104. E : Et il a mémorisé le résultat c'est ça ?
105. P : Oui, ça y est. Le résultat est mémorisé dans Alpro.
106. E : Oui, oui. Donc on refait A plus B fois 2.
107. P : Alors A plus B fois 2.
108. E : Et on fait égale. AC/ON. Et donc on refait Ans fois Ans plus 1.
109. P : Oui, on fait égale.
110. E : Et on a le résultat. Et on refait ça, on multiplie par 3. En fait, à chaque fois on multiplie par 3, par 4…
111. P : D'accord, est-ce que vous êtes d'accord que le programme est correct ou non ? Et que ça imprime tout ce qui est demandé ? A chaque fois que j'appuie sur la touche égale, j'imprime ça car je suis CALCULATOR, est ce que vous pensez que j'imprime toutes les images et seulement les images ? Oui, ça va ? Le programme est correct ?
112. Es : Oui.
113. P : Ets ce que le programme est un peu long ?
114. Es : Si, si.
115. P : Si, par ce que vous n'avez pas encore écrit totalement tout le programme ?
116. E (au tableau) Si, si.
117. P : Ah, non, non. Parce que ce que je veux dire, pour l'instant je vois (il pointe le doigt sur leur production) un deux trois…une trentaine là. Il n'est pas encore total.
118. E (au tableau) : Parce que je fais m.
119. P : Ah, il n'est pas encore fini le programme ? (il continue de taper) Mais m, m c'est quoi ? Mais le CALCULATOR, m qu'est ce que c'est ?
120. E (au tableau) : C'est un nom.
121. P : Non, non, là CALCULATOR il sait appuyer sur les touches.
122. E : Il appuie n fois 1, 2, 3, 4, 5, 6, 7, 8, 9 jusqu'à l'infini.
123. P : Jusqu'à l'infini ?
124. Es : Non, non.
125. E (du binôme) : Non jusqu'à la fin de l'intervalle.

Le programme en acte de l'enseignant- CALCULATOR sous la dictée du binôme :

3 Sto A AC/ON 0.03 Sto B AC/ON A + B = AC/ON Ans × ÷ C/OFF Ans + 1 = [9,8209] Ans × Ans + 1 = [97,45007681] A + B × 2 = [-2,94] AC/ON Ans × Ans + 1 = [9,6436]

126. P : On a quel problème là ?
127. E : La répétition.
128. P : Oui, la répétition et quoi ? Car vous dites il faut aller jusqu'à l'infini et d'autres disent non. Alors, c'est…
129. E : C'est l'intervalle.
130. P : Mais du point de vue du programme c'est…
131. E (un autre de la classe) : C'est le problème de finition.
132. P : Pardon c'est…

[1] Ce binôme ajoute « AC/ON » à chaque fois alors que cette touche n'est pas écrit dans le message.

133. E : Finition.
134. P : Oui, finition ce n'est peut – être pas le bon mot mais c'est le problème, le problème de…le point d'arrêt, non ? Il faut qu'on s'arrête, non ? Bon, ça va vous amener à un nouveau problème. Parce que finalement on s'aperçoit d'une chose que toutes les actions à CALCULATOR sont écrites sous formes de touches. Donc ce qu'on aimerait savoir c'est il y a combien de touches ?
135. Es : C'est long.
136. P : C'est long, ah oui d'accord c'est long mais c'est combien, long ? (il donne la nouvelle fiche à chaque binôme) (…)

La simulation des capacités du Robot par l'enseignant oblige le binôme à une mémoire supplémentaire « *Bah, on fait égale et stocké* » (100). Cependant cette instruction n'est pas effectuée par l'enseignant alors qu'il dit « *Oui, ça y est. Le résultat est mémorisé dans Alpro* » (105) pour répondre à la question « *et il a mémorisé le résultat c'est ça ?* » (104) du binôme.

Cette simulation a abouti à un nouveau problème, celui de la phase 3 : *« ça va vous amener à un nouveau problème. Parce que finalement on s'aperçoit d'une chose que toutes les actions à CALCULATOR sont écrites sous formes de touches. Donc ce qu'on aimerait savoir c'est il y a combien de touches* ? » (134).

III. Phase 3 : Coût d'un programme écrit en langage Alpro à la machine

Nous rappelons l'énoncé de cette phase[1].

Phase 3 (15 minutes) Travail en binôme avec papier, stylo et Alpro

La longueur du message à CALCULATOR est égale au nombre d'appuis de CALCULATOR sur Alpro.
Combien de fois en tout CALCULATOR appuie-t-il sur les touches de la calculatrice Alpro ?

Nombre de fois : ☐

Expliquez comment vous avez calculé ce nombre :

Comment les binômes ont-ils calculé le nombre d'appuis de touches ?
Deux procédures de calculs sont présentes.

- Procédure 1 : Calculer le nombre d'appuis de touche pour l'invariant puis le multiplier par le nombre de répétition ;
- Procédure 2 : Partager les touches en deux groupes, dénombrer le nombre d'appuis de touches pour chaque groupe puis faire la somme de deux nombres obtenus.

Le tableau 4 ci-après donne deux exemples de ces procédures ainsi que les effectifs pour chaque procédure.

Procédure 1	Procédure 2
Nombre de fois 3652 Nombre de fois que Calculator a appuyé sur Alpro pour le calcul précédent × $\frac{\lVert -3 \rVert + 2}{0{,}03} = \underbrace{16666}...$ $22 \times 166 =$ (1^{er} F)	Nombre d'appuis de touches : 2486 Pour donner des ordres à CALCULATOR, on écrit - 3 Sto A AC/On A × A + 1 = AC/On (*) A + 0,03 Sto A A × A + 1 = AC/On (**) Répéter 164 fois (*) il y a 11 touches à appuyer (**) il y a 15 touches à appuyer ⇒ il faut appuyer (15 × 165) + 11 = 2486 (1^{er} V)
Nombre de binômes : 9 dont 5 (1^{er} F), 4 (1^{er} V)	Nombre de binômes : 7 dont 4 (1^{er} F), 3 (1^{er} V)

Tableau 4. Résultats sur le nombre d'appuis de touches

[1] Rappelons que dans cette phase, il reste seulement 8 binômes pour la classe 1^{er} V.

Nous donnons un autre exemple dans l'institution vietnamienne :

> Nombre d'appuis de touches : 2490
> Pour déterminer un nombre, le robot doit appuyer sur 15 touches. Dans l'intervalle [- 3 ; 2], avec l'écart égal à 0,03 on doit déterminer 166 nombres. Donc le robot doit réaliser au total 166 × 15 = 2490 fois d'appuis de touches. (1er V)

Le tableau 5 ci-après montre que les explications des binômes pour leur calcul du nombre d'appuis de touches intègrent le nombre de répétitions du calcul, et donc la condition d'arrêt de la répétition. La formulation de cette condition d'arrêt revient à un dénombrement délicat, celui des nombres x_i appartenant à l'intervalle [a ; b] (analyse *a priori*).
Pour les binômes qui font la division 5 ÷ 0,03 pour déterminer le nombre de répétition, le résultat est 166 comme le montre l'exemple ci-dessus. Un seul binôme (1er F) donne un résultat correct : « 17 + 13 × 5 / 0,03 = 2175 » ; pour tous les autres, le premier calcul (image de – 3 par f) n'est pas ajouté à 166.
La notation 167(5) signifie que 5 binômes donne le résultat 167.

1er F	1er V
100(1), **166(6)**, 167(1), 200(1), pas d'explication (1)	**164(4)**, 166(2), 165(1), pas d'explication (1)

Tableau 5. Explication du calcul du nombre d'appuis de touches

Dans la classe 1er F:

- le nombre 200 est le résultat d'une erreur de calcul sur la longueur de l'intervalle, 6 au lieu de 5 : « 6 ÷ 0,03 = 200 ; 15 × 200 = 3000 »,
- le nombre 100 est le résultat « 1 + 99 ». On suppose que ce binôme trouve 99 répétitions puis ajoute 1 pour l'étape qu'il qualifie de « réparation ».

IV. Phase 4 : Amélioration du robot CALCULATOR

Nous rappelons l'énoncé de cette phase.

Phase 4 (15 minutes)
Travail en binôme avec papier, stylo et Alpro (10 minutes)
On va essayer dans la séance suivante d'améliorer le robot CALCULATOR pour rendre la longueur du message inférieure au nombre d'appuis de CALCULATOR sur Alpro. Cela nous permettra non seulement d'écrire tout le message mais aussi de le rendre considérablement plus court.
Pour améliorer le robot CALCULATOR, que proposez-vous ?
Écrire vos propositions :

Les propositions faites par des binômes portent non seulement sur le robot CALCULATOR mais aussi sur Alpro, certains binômes proposant même des améliorations d'Alpro et de CALCULATOR.
Les propositions « imprécises »[1] ne sont pas présentées ici. Cependant, deux d'entre elles portent sur la mémoire. Trois élèves de 1er F ne donnent pas de réponse.

Les autres propositions sont récapitulées dans le tableau 6 ci-après :

[1] Ce sont des propositions : améliorer la mémoire ; mémoriser ce qui a été fait ; ajouter quelques touches ; que ça décline toutes les images lorsqu'on appuie sur [=] successivement ; comprendre la voix humaine ;

	Propositions sur Alpro	Propositions sur CALCULATOR
Classe 1er V	- Ajouter la touche x^2 (2) ; - Ajouter les parenthèses ; - Ajouter une touche pour mémoriser la formule A × A + 1 ;	1) *Exécuter les étapes automatiquement d'une manière répétitive* ; 2) *Avoir la capacité de répéter des calculs contenus dans le programme* ; 3) Avoir la capacité de remplacer dans l'opération précédente par un autre nombre ; 4) Savoir remplacer un nombre dans une opération faite précédemment ; 5) Capacité d'appuyer sur plusieurs touches ;
Classe 1er F	- Puissance de 10 (2) ; - Touche x^2 ; - La possibilité de storer une expression littérale (ex : $x^2 + 1$) ; - Calcul de fonction ;	1) *La possibilité de refaire un procédé n fois (loop)* ; 2) Etant donné une A = 0,03 est toujours × 2 pour chaque image, CALCULATOR *devrait s'habituer au bout de 2 calculs*. Il suffite plus de faire = pour afficher les prochains images (comme Excel) ;

Tableau 6. Des propositions faites par des binômes

Les améliorations sur Alpro ont essentiellement pour but d'économiser le nombre d'appuis de touches. Par exemple, la touche « ^2 », ou « x^y ». Deux binômes portent sur la touche « puissance 10 ». Nous ne pouvons pas interpréter ce rajout. L'amélioration sur la touche pouvant mémoriser la formule A × A + 1 (ou x^2 + 1) revient à fournir, soit une mémoire expression (comme la calculatrice Casio) ou une mémoire de formule (la calculatrice TI, cf. chapitre C2), soit la capacité de « programmer une fonction »[1], la fonction x^2 + 1 en l'occurrence. Cependant si une calculatrice possède l'une ou l'autre de ces deux capacités, elle donne seulement *une image de fonction* par chaque manipulation. Il faut entrer le nombre v dans la calculatrice puis appeler la fonction déjà « programmé » pour obtenir l'image f(v). Cette proposition (3 et 4) pour améliorer robot CALCULATOR est faite par deux binômes.
Comme nous l'avons prévu dans l'analyse *a priori* aucun binôme n'a proposé de type d'amélioration sur la capacité de comprendre des expressions logiques.
Seules trois propositions portent (1 en 1er F et 2 en 1er V) sur la capacité de pouvoir répéter l'exécution d'un groupe de touches du robot CALCULATOR. Le binôme en 1er F n'a utilisé aucun signe pour exprimer la répétition dans la phase 1. L'un des deux binômes en 1er V a utilisé un signe « … » et l'autre le mot « répéter » pour exprimer la répétition.
Malgré le calcul de nombre d'appuis de touches (dans la phase 3) qui donne un nombre assez grand (au minimum 1090 en 1er F) la tâche d'améliorer les capacités de CALCULATOR en lui attribuant celle de répétition est problématique pour des élèves.

V. Synthèse par l'enseignant des propositions d'amélioration de CALCULATOR

Dans cette synthèse, en s'appuyant sur les propositions des binômes, l'enseignant institutionnalise l'idée d'un robot amélioré, CALCULATOR II, qui, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution d'un groupe de touches d'Alpro.
Nous donnons ci-après les chroniques de cette synthèse dans les deux classes.

V.1. Synthèse dans la classe de 1er F

L'enseignant commence par préciser que l'on ne cherche pas à améliorer la calculatrice Alpro mais le robot CALCULATOR, qui se sert d'Alpro. Puis il rappelle la longueur du message à CALCULATOR, qui est égale au nombre d'appuis de touche (entre 2000 et 4000 touches !).

[1] C'est le nom dans EMS en France pour désigner cette activité. Elle consiste à entrer une fonction f dans la mémoire de la calculatrice et de sortir l'image f(v) du nombre v tapé à partir de l'écran de calculatrice.

1. P : […] Je vous rappelle aussi que dans la dernière séance nous avons fait un certain travail. On vous a demandé des propositions pour améliorer CALCULATOR et j'avais insisté sur le fait que ce robot CALCULATOR n'est pas Alpro. Le robot se sert d'Alpro. Il appuyait sur les touches d'Alpro. Il est donc différent d'Alpro. Il a la capacité d'exécuter les touches d'Alpro quand on le lui demande et d'imprimer automatiquement. Je vous ai demandé des propositions pour améliorer le robot CALCULATOR en tenant compte du fait que nous nous sommes aperçus que le message adressé à CALCULATOR comportant une longueur qui est autant de nombres de touches que CALCULATOR doit appuyer. Et nous l'avons calculé. Je pense que certains, il y a certains qui ont déjà calculé ce nombre. Deux milles et voir quatre milles touches. C'est énorme. Donc un message qui comporte 4 milles touches. On se rend compte qu'il y a quelque chose à faire. Alors vos propositions, et j'ai lu votre message et je m'aperçois que […] Par exemple celui là. (Il lit le message) « que ça décline toutes les images lorsqu'on appuie sur la touche « = » successivement » […] Et ceci « savoir des intervalles de fonctions » ? […] Ce que je voulais dire, en regardant les messages je me suis aperçu que pour beaucoup d'entre vous, les propositions portaient sur Alpro.

Après avoir disqualifié encore une fois les propositions d'amélioration d'Alpro, l'enseignant passe aux propositions portant sur le robot CALCULATOR.

2. P (suite) : Or notre demande c'était d'améliorer CALCULATOR. J'ai vu quand même ceux qui parlaient de CALCULATOR. Je vois celle-là (il lit le message) « Ètant donnée que A = 0.03 est toujours x 2 pour chaque image, CALCULATOR devrait s'habituer au bout de 2 calculs ». S'habituer au bout de 2 calculs ? Je me suis dit quand même que ce message s'adressait à CALCULATOR. C'est une capacité de CALCULATOR. Est-ce qu'on peut formuler un peu plus précisément ? Ca veut dire quoi « s'habituer » ?
3. E1 : C'est savoir répéter les mêmes…
4. P : Les mêmes ?
5. E1 : Les mêmes, les procédures.
6. P : Les mêmes procédures mais…d'accord ?
7. Es : (silence)
8. P : Est-ce qu'on est d'accord là-dessus ? Ou pas ? (en s'adressant à l'élève qui vient de prendre la parole) C'est vous qui avez fait ça ? Non. Je ne sais pas mais est-ce que celui qui a écrit ça peut nous dire…Est-ce que c'est une bonne traduction ou pas ?
9. E2 (auteur de la proposition, timidement) : Qui refait les modifications qui …
10. P (il répète) : Ca va refaire les modifications qui ?
11. E2 : ont été faites.
12. P : Qui ont été faites. On sent quand même qu'il y a l'idée de reprendre les procédures, je ne sais pas quoi, mais de les modifier de telle manière à assurer la répétition, oui ou pas ? Ou c'est pas ça son idée ? Il faut qu'on en discute un petit peu si…Oui, **assurer la répétition c'est quand même la question de base** de ce que vous vous faites à la main. En fait, vous faites bien les répétitions et vous modifiez des choses pour faire la répétition. J'aimerais qu'on se penche sur cette idée **et qu'on sache que s'il y a des choses à répéter, qu'est ce qu'il y a à répéter**. **Qu'est ce qu'on va demander à CALCULATOR de répéter en n'oubliant pas que l'on va lui demander. D'abord, on aimerait savoir ce que CALCULATOR doit répéter**. C'est la question à laquelle j'aimerais que vous répondiez. Je vous distribue les petites fiches. […] Voilà, on veut assurer la répétition et on sait que cette répétition devrait permettre de diminuer le nombre de touches à demander à CALCULATOR d'appuyer. Car on a déjà vu et on sait que c'est un grand nombre. On décide donc d'améliorer…On vous propose de prévoir que CALCULATOR puisse répéter l'exécution d'un groupe de touches d'Alpro. […]

Pour l'enseignant « *assurer la répétition c'est quand même la question de base* » et il désigne aux élèves l'enjeu de la situation suivante : « *J'aimerais qu'on se penche sur cette idée et qu'on sache que s'il y a des choses à répéter, qu'est ce qu'il y a à répéter* », la justification de cette formulation étant de « *diminuer le nombre de touches à demander à CALCULATOR d'appuyer* » et donc la longueur du programme.

V.2. Synthèse dans la classe de 1er V

Comme dans la classe de 1e F, l'enseignant commence par faire le tri, avec la collaboration des élèves, entre les propositions d'amélioration d'Alpro et celle du robot CALCULATOR. .

300. P : Je vais passer en revue vos propositions. Par exemple, il y a un binôme qui a écrit « Il y a une touche pour enregistrer A × A + 1 et il suffit que le robot calcule la valeur de x enregistré dans A et le remplace dans la formule précédente. » Alors, c'est Alpro que vous améliorez pas CALCULATOR.

301. E1 (du binôme qui écrit la proposition) : Oh, c'est CALCULATOR ? Oh, on s'est trompé.
302. P : Ou un autre binôme « Avoir la capacité de répéter des calculs contenus dans le programme ». Donc c'est la capacité de ?
303. Es : Alpro.
304. P : C'est la capacité de CALCULATOR qui peut répéter les messages. Un autre binôme qui propose d'ajouter « ajouter la touche carré et parenthèses » mais ça c'est de ?
305. Es (ils rient) : Alpro.
306. P : Mais ce groupe a proposé aussi une amélioration de CALCULATOR qui est « Exécuter les étapes et les répéter automatiquement ». Cela est-il correct ?
307. Es : Oui.
308. P : Oui, ou un autre exemple « Exécuter automatiquement les étapes répétitives ». Vous voyez c'est pour le robot CALCULATOR. Et ce même groupe « la capacité de remplacer un nombre dans une opération avant par un autre et comprendre la voix humaine ».
309. E1 : Ce qui est impossible.
310. P : Ou un autre groupe « ajouter certaines touches ; utiliser le clavier et non pas la souris ; mémoriser ce qui est déjà fait »
311. E2 : C'est trop général.

Il ne fait aucun doute pour les élèves que Alpro peut mémoriser :

312. P : Ca, vous voyez, est ce qu'on peut mémoriser des choses avec Alpro ?
313. Es : Oui.
314. P : Par quelle touches ?
315. Es : A, B, C.

L'enseignant conclut sur l'amélioration du robot « *ce robot doit pouvoir répéter plusieurs fois un groupe de touches* ». Il donne un objectif « concret » à la situation suivante : « *ce robot doit pouvoir répéter plusieurs fois un groupe de touches* […] *On va voir quels sont les groupes de touches à répéter* », mais contrairement à l'enseignant précédent il ne justifie pas cet objectif par la diminution de la longueur du programme à adresser au robot.

316. P : Vous avez ainsi vu que les propositions que vous avez faites à CALCULATOR c'est que **ce robot doit pouvoir répéter plusieurs fois un groupe de touches**. Alors dans la phase suivante, on essaie d'être plus concret. On va voir **quels sont les groupes de touches à répéter** et on va le proposer à CALCULATOR et avec la nouvelle capacité que vous avez lui proposé qui est de pouvoir répéter les actions, il sera en mesure de calculer les images. La question est donc « Ecrire un groupe de touches à répéter ». Donc quels sont les groupes de touches ?

VI. Conclusion de l'analyse *a posteriori* de la situation 2

Les résultats de l'analyse de cette situation montrent que dans les premiers programmes écrits de calcul répétitifs, la majorité des élèves délèguent au robot CALCULATOR la prise en charge de la répétition et de l'arrêt des calculs : ils ont donc pour la plupart écrit un programme « impossible » au robot CALCULATOR qui ne sait ni répéter ni arrêter des calculs.

L'enseignant a du intervenir pour rappeler les capacités et les limitations du robot CALCULATOR.

Quand les mémoires variables A, B, C sont utilisées, elles sont initialisées au début de l'itération. Ce résultat infirme des études précédentes qui montraient que cette opération fondamentale étaient problématique « *même pour des élèves d'un plus haut niveau de connaissance dans la construction des boucles* » (Samurçay 1985, cité dans Méjias 1985, p. 97). La maîtrise de cette opération, observable de l'instrumentation des touches mémoires variables A, B et C, continue à se forger dans cette situation et dans le prolongement des calculs mis en place dans la situation 1.

Les résultats de l'expérimentation valident une partie de notre hypothèse : l'écriture d'un programme de calcul répétitif à une machine à mémoire effaçable est une condition favorable à l'émergence de la notion variable comme mémoire effaçable. Presque la moitié des élèves effectue la mise à jour des variables, observable de l'effaçabilité d'une mémoire déjà initialisée.

Cependant un nombre quasi équivalent d'élèves continue à concevoir les mémoires variables A, B, C comme des variables mathématiques. Ce statut des mémoires variables A, B, C présent dès la situation 1, résiste aux conditions mises en place, en particulier à la répétition importante des calculs. Ils ont recours à des palliatifs comme la mémoire papier ou des symboles ou mots habituels de la répétition.

Le dénombrement du nombre d'appuis de touches exige d'identifier l'invariant de la répétition du calcul puis de calculer le nombre de répétition de cet invariant, préalable à la formulation d'une condition d'arrêt. Ce dénombrement est une tâche problématique puisque un seul binôme donne un résultat correct.

Les deux algorithmes de calcul sont disponibles, l'algorithme de proche en proche M(R) et l'algorithme utilisant un modèle fonctionnelle M(F) comme le montrent les calculs dans le premier problème de tabulation.
Le travail d'écriture d'un programme en langage Alpro est un travail de transformation des algorithmes : cette transformation est plus coûteuse en mémoires et en nombre d'opérations pour l'algorithme M(F) que pour l'algorithme M(R). Ce coût apporte une explication à la prédominance de l'algorithme M(R) dans l'écriture des programmes observés. Mais même dans le cas de l'algorithme M(R), le travail d'écriture d'un programme reste difficile puisque seul un binôme réussit l'écriture de l'invariant en langage Alpro.
Pour l'algorithme M(F), une difficulté supplémentaire s'ajoute au coût, celle de la formulation de la « variable compteur », variable à construire car absente de l'énoncé du problème : elle n'est explicitée dans aucun des programmes M(F), les élèves utilisant la mémoire papier pour noter les premières valeurs de cette variable et attribuant au robot CALCULATOR la capacité de continuer le calcul de ces valeurs.

La situation 2 s'achève sur le besoin d'améliorer le robot CALCULATOR puisque la longueur du programme est égale au nombre d'appuis de touches, calculé dans la phase 3 : l'écriture du programme dans un temps limité est donc impossible !
L'institutionnalisation de l'enseignant à la fin de la situation 2 est nécessaire pour que l'enjeu soit bien l'amélioration du robot CALCULATOR et non de la calculatrice que les élèves cherchent à rapprocher des calculatrices qu'ils connaissent.

L'amélioration du robot revient à concevoir une unité de commande dans la machine (Alpro, CALCULATOR) sans changer l'unité de calcul.

Chapitre 3

Première partie
Analyse *a priori* de la situation 3
Co-évolution du langage et de la machine (Alpro, CALCULATOR II)
Machine ordinateur de Von Neumann

Introduction

La situation 3 continue le processus initié à la fin de la situation 2 (phase 4) : l'amélioration du Robot CALCULATOR qui elle-même fait évoluer le langage et la machine.

Nous avons choisi d'améliorer le Robot CALCULATOR en lui donnant la capacité de « comprendre » *un nombre limité de mots* exprimant la répétition. Et donc nous avons éliminé l'autre amélioration (compréhension d'expressions logiques) beaucoup trop complexe à mettre en place dans le cadre de notre projet d'une ingénierie à durée limitée.
La capacité de pouvoir répéter l'exécution d'un calcul sont liées intrinsèquement à la fonction de traduction d'un programme : le Robot CALCULATOR II peut traduire un programme écrit comportant des touches d'Alpro et des *mots* exprimant la répétition. Il le traduit en un langage qu'il peut exécuter directement sur Alpro. Dans ce sens, le Robot CALCULATOR II est un compilateur qui enregistre le programme écrit (« programme enregistré » de la machine ordinateur) et le traduit pour l'exécuter. Dans la situation 3, la machine (Alpro, CALCULATOR II) est une machine ordinateur suivant l'architecture de Von Neumann (cf. chapitre B2).
Ce travail sur le Robot CALCULATOR fait donc participer les élèves à la fabrication d'une nouvelle machine

L'évolution de la machine va de pair avec une évolution du langage de la programmation compris par la machine : le langage évolué à la machine améliorée (Alpro, CALCULATOR II) comprend une suite de touches d'Alpro et *quelques mots*.
Cette situation permet ainsi l'émergence en actes d'éléments fondamentaux d'un langage évolué comme :

- la séquentialité : déjà présente dans le langage Alpro
- l'itération : nouvelle structure du langage destiné à (Alpro, CALCULATOR II)

Rappelons l'hypothèse formulée à la fin de la partie B et qui fonde la conception de cette situation.

Hypothèse

L'écriture d'un message à une machine de Von Neumann pour obtenir l'exécution d'un algorithme itératif de calcul est une condition favorable à l'émergence en actes d'éléments d'un langage évolué : séquentialité, branchement et itération.

I. Situation 3 – Phase 1 : explicitation de l'invariant de boucle

L'enjeu de cette phase est la formulation de l'invariant de la répétition des calculs *en langage Alpro*, c'est-à-dire un groupe de touches sur lesquelles le Robot CALCULATOR aura à appuyer de manière répétitive pour exécuter le calcul. Cet invariant du calcul a déjà été travaillé implicitement dans la situation 2, au cours de la phase 3 du dénombrement des touches d'appuis d'Alpro par le Robot CALCULATOR.

I.1. Énoncé de la phase 1 de la situation 3

Phase 1 (15 minutes)
Travail en binôme avec papier, stylo et Alpro
Pour rendre la longueur du message inférieure au nombre d'appuis de CALCULATOR sur Alpro, on décide d'augmenter les capacités de CALCULATOR.
En plus des capacités de CALCULATOR, le robot amélioré, CALCULATOR II doit pouvoir répéter l'exécution d'un groupe de touches d'Alpro.
Écrire un groupe de touche à répéter, en vous assurant sur Alpro que la répétition fournit bien les nombres attendus.
Groupe de touches à répéter :

I.2. L'écriture en langage Alpro de l'invariant du calcul répétitif à CALCULATOR II

L'écriture de l'invariant du calcul répétitif en langage Alpro au Robot CALCULATOR II (qui peut les répéter) peut conduire à quatre groupes de touches en relations aux deux algorithmes M(R) ou M(F) (cf. chapitre D2).

Algorithme M(R)		**Algorithme M(F)**	
Mémoires A, B, C	Mémoires A, B, C et Ans	Mémoires A, B, C	Mémoires A, B, C et Ans
Initialisation de la mémoire : [- 3] [Sto] [A] **Groupe de touches à répéter :** [A] [×] [A] [+] [1] [=] [A] [+] [0,03] [Sto] [A]	**Initialisation de la mémoire** : [- 3] [Sto] [A] **Groupe de touches à répéter :** [Ans] [×] [Ans] [+] [1] [=] [A] [+] [0,03] [Sto] [A]	**Initialisation de la mémoire** : [0] [Sto] [A] **Groupe de touches à répéter :** [-3] [+] [A] [×] [0,03] [Sto] [B] [B] [×] [B] [+] [1] [=] [A] [+] [1] [Sto] [A]	**Initialisation de la mémoire** : [0] [Sto] [A] **Groupe de touches à répéter :** [-3] [+] [A] [×] [0,03] [Sto] [B] [Ans] [×] [Ans] [+] [1] [=] [A] [+] [1] [Sto] [A]

Tableau 1. Groupes avec mémoires variables de touches attendus suivant les deux algorithmes M(R) et M(F)

La consigne suggère aux élèves de valider les groupes de touches qu'ils ont écrit en jouant le rôle du Robot : « *Écrire un groupe de touche à répéter, en vous assurant sur Alpro que la répétition fournit bien les nombres attendus.* »

La validation publique des groupes de touches proposés s'effectue dans la phase suivante.

II. Situation 3 – Phase 2 : institutionnalisation des groupes de touches

À la fin du temps imparti, l'enseignant récapitule les groupes de touches proposés dans la classe (groupe avec mémoires Ans, groupe avec mémoires variables).
Pour chaque proposition différente, l'enseignant joue le rôle du robot fictif CALCULATOR II en exécutant « à la main » les groupes de touches proposés sur Alpro publique : rétro-projection au tableau.

Cette validation effective à partir de certaines réponses des binômes, oblige à débuter l'exécution d'un groupe de touches sur Alpro, et donc débouche *sur l'initialisation explicite de la variable informatique (mémoires A, B, C)* et sur l'arrêt effectif et implicite de l'exécution
Cette exécution fournit des éléments de validation du groupe de touches produit par l'élève.
Cette phase débouche sur l'institutionnalisation de deux groupes de touches liés aux algorithmes M(R) et M(F) : ces deux groupes restant écrits au tableau pour la phase qui suit.
Ce sont les groupes suivants :

Algorithme M(R)	Algorithme M(F)
A x A + 1 = A + 0 , 0 3 STO A	- 3 + A x 0 , 0 3 STO B B x B + 1 = A + 1 STO A

Tableau 2. Groupes de touches en relation avec les deux algorithmes M(R) et M(F) laissés au tableau

III. Situation 3 – Phase 3 : comment « écrire » à CALCULATOR II la répétition ?

L'enjeu de cette phase est de faire évoluer le langage Alpro en un langage (Alpro, CALCULATOR II) comprenant une suite de touches d'Alpro et *quelques mots* que CALCULATOR II « comprend » et qui permet d'écrire un message le plus court possible à la machine

III.1. Enoncé de la phase 3 de la situation 3

Phase 3 (15 minutes)
Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR II, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.
Comment le lui dire ?
On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.
Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II
CALCULATOR II, en exécutant ce message, doit imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$
Message :
Longueur du message : ☐
Pour calculer la longueur du message, les mots seront comptés comme les touches.
Liste de mots :

III.2. Les mots et les programmes possibles

Les mots nécessaires sont demandés chronologiquement *après* l'écriture du message à la machine. En effet nous faisons l'hypothèse que ces mots sont trouvés en interaction avec l'écriture d'un message qui doit être le plus court possible, sachant qu'un mot compte comme une touche[1].

Ces « mots » construisent un répertoire commun à CALCULATOR II - Unité de commande de la machine (Alpro, CALCULATOR II) - et au binôme qui écrit le message.

On peut envisager au moins :

[1] Un mot ou un groupe de mots (instruction) ne peut être en réalité compté comme une touche : il faut lui attribuer un poids puisque ce mot (ou ce groupe de mots) représente un ensemble complexe d'opérations « machines ».

- un mot (ou un groupe de mots), comme « répéter », qui sont un ordre au robot CALCULATOR II d'appuyer sur la suite des touches qui vient juste après le mot ;
- un mot (ou un groupe de mots), comme « fin », qui délimite le groupe de touches à répéter ;
- un mot (ou un groupe de mots), qui est un ordre pour arrêter la répétition.

Pour être effectif, un programme écrit dans ce langage évolué utilise une structure d'itération (programme à boucle), comprenant :
- l'initialisation des variables ;
- la mise à jour des variables ;
- une condition d'arrêt.

Nous donnons les exemples de deux programmes à boucle[1] correspondants aux deux algorithmes M(R) et M(F) :

	Algorithme M(R) $x_{n+1} = x_n + p$	**Algorithme M(F)** $x_{n+1} = a + np$
Programmes	- 3 Sto A Répéter 167 fois A × A + 1 = A + 0,03 Sto A Fin	Répéter 167 fois 0 Sto A – 3 + 0,03 × A Sto B B × B + 1 A + 1 Sto A Fin
Coût de l'écriture	*24 touches* *3 mots : répéter, fois, fin*	*30 touches* *3 mots : répéter, fois, fin*

Tableau 1. Deux programmes en langage évolué à la machine (Alpro, CALCULATOR II)

On peut noter que la longueur des messages est réduite considérablement de 3600 appuis de touches (au maximum pour des programmes sans boucle) à 30 touches (au maximum pour des programmes avec boucle).

IV. Organisation d'un débat « scientifique » sur la justification et la complexité des algorithmes des programmes pour résoudre le problème mathématique

Qu'est-ce qui dans le programme justifie que l'on a obtenu *toutes les images* par la fonction f des valeurs de x de l'intervalle [-3 ; 2] *et seulement ces images* [f : x → f(x) = $x^2 + 1$] ?
Quel programme exige le moins grand nombre d'appuis de touches, le moins grand nombre d'opérations, le moins grand nombre de mémoires ?

Chaque binôme écrit sur une affiche son programme (papier de grand format) pour le présenter à l'ensemble de la classe.
L'enseignant organise le débat en choisissant deux programmes : ces programmes sont débattus relativement à la question posée, successivement et dans l'ordre :
En ce qui concerne les opérations sur les variables :
- un programme sans initialisation ;
- un programme sans mise à jour ;
- un programme sans condition d'arrêt.

En ce qui concerne les algorithmes :
- un programme M(R) ;

[1] Dans l'annexe D3, nous présentons tous les programmes s'appuyant sur les deux algorithmes M(R) et M(F) avec 4 types de boucles

- un programme M(F) ;

Chaque présentation est commentée et suivie d'un débat sur la justification et la complexité des algorithmes sous-jacents aux programmes et débouche sur un travail coopératif d'amélioration du programme.

Pour finir la séance 3, l'enseignant conduit une réflexion sur les « mots » du répertoire commun à CALCULATOR II et à l'émetteur qui est devenu élève - programmeur.

V. Conclusion de l'analyse *a priori* de la situation 3

L'aboutissement de l'ingénierie didactique est l'écriture des messages à la machine (Alpro, CALCULATOR II), ordinateur de Von Neumann. Les programmes obtenus dans cette situation permettent d'avoir un certain nombre d'observables sur l'émergence :

- des notions de variable à l'intérieur de boucles de programmes plus ou moins abouties
- d'éléments du langage évolué pour les écrire.

Le débat scientifique sur la justification et le coût des algorithmes utilisés dans le programme permet un retour réflexif sur les notions mathématiques telles que intervalle, images d'un nombre, suites numériques.

Que s'est-il passé ?
Nous allons maintenant analyser les réalisations de la situation 3 dans EMS en France et au Viêt-nam pour répondre à cette question.

Chapitre 3

Deuxième partie
Analyse *a posteriori* de la situation 3

Dans cette situation, le processus initié à la fin de la situation 2 (phase 4), à savoir l'amélioration du Robot CALCULATOR, continue. Rappelons que cette amélioration intègre une évolution du langage et de la machine : l'enjeu sera alors d'écrire un programme informatique dans le langage évolué à la machine ordinateur de Von Neumann.
Les conditions du recueil des données sont les mêmes que dans la situation 2 :

- Les élèves travaillent en binômes sur Alpro durant toutes les phases de la situation. On peut donc reconstituer leur programme en acte à partir des fichiers historiques
- Les réponses écrites et les brouillons sont récupérés
- Les deux binômes choisis (un dans chaque institution) sont enregistrés (enregistrements vidéo pour l'un en 1er F et audio pour l'autre en 1er F)
- Les deux enseignants sont filmés.

I. Situation 3 – Phase 1 : explicitation de l'invariant de boucle

L'enjeu de cette première phase est d'écrire en langage Alpro l'invariant de la répétition dans le calcul de la tabulation de f, c'est-à-dire ce que le robot CALCULATOR II (amélioré par rapport à CALCULATOR) doit pouvoir répéter, de façon à diminuer drastiquement la longueur du programme.

Rappelons l'énoncé de cette phase.

Phase 1 (15 minutes)
Travail en binôme avec papier, stylo et Alpro
Pour rendre la longueur du message inférieure au nombre d'appuis de CALCULATOR sur Alpro, on décide d'augmenter les capacités de CALCULATOR.
En plus des capacités de CALCULATOR, le robot amélioré, CALCULATOR II doit pouvoir répéter l'exécution d'un groupe de touches d'Alpro.
Écrire un groupe de touche à répéter, en vous assurant sur Alpro que la répétition fournit bien les nombres attendus.
Groupe de touches à répéter :

Les groupes de touches attendus sont donnés dans le tableau 1.

Algorithme M(R)		**Algorithme M(F)**	
Mémoires A, B, C	Mémoires A, B, C et Ans	Mémoires A, B, C	Mémoires A, B, C et Ans
Initialisation de la mémoire : [- 3] [Sto] [A] **Groupe de touches à répéter :** [A] [×] [A] [+] [1] [=] [A] [+] [0,03] [Sto] [A]	**Initialisation de la mémoire** : [- 3] [Sto] [A] **Groupe de touches à répéter :** [Ans] [×] [Ans] [+] [1] [=] [A] [+] [0,03] [Sto] [A]	**Initialisation de la mémoire** : [0] [Sto] [A] **Groupe de touches à répéter :** [-3] [+] [A] [×] [0,03] [Sto] [B] [B] [×] [B] [+] [1] [=] [A] [+] [1] [Sto] [A]	**Initialisation de la mémoire** : [0] [Sto] [A] **Groupe de touches à répéter :** [-3] [+] [A] [×] [0,03] [Sto] [B] [Ans] [×] [Ans] [+] [1] [=] [A] [+] [1] [Sto] [A]

Tableau 1. Groupes de touches que CALCULATOR II doit pouvoir répéter selon l'algorithme de calcul

Dix-sept propositions de groupes de touches (sur 18 binômes présents) ont été écrites : 9 en 1er F et 8 en 1er V.
Le tableau 2 ci-après donne la répartition des propositions selon les groupes de touches associés aux deux algorithmes M(R) et M(F). Nous préciserons la signification de la catégorie « autres » par la suite.
Ces groupes de touches représentent des invariants de la répétition dans les programmes adressés à la machine (Alpro, CALCULATOR II), c'est-à-dire chacun le corps d'une boucle de ce programme.
Une première écriture de l'invariant a été réalisée dans la phase 2 de la situation lors de l'écriture en langage Alpro de programmes à (Alpro, CALCULATOR).

Afin de pouvoir comparer l'évolution dans l'écriture en langage Alpro de cet invariant, nous indiquons entre parenthèses les effectifs dans la phase 2 (situation 2) pour un groupe de touches *utilisant les mêmes algorithmes*.

Algorithmes / Classes	Algorithmes M(R)	Algorithmes M(R)'	Algorithmes M(F)	Algorithmes M(F)'	Autres	Non réponse	Total
1er F	4 (4)	0 (2)	3 (2)	2 (2)	0 (0)	1 (0)	10
1er V	3 (3)	0 (2)	2 (1)	0 (0)	**3** (1)	0 (2)	8
Total	7 (7)	0 (4)	5 (3)	2 (2)	3 (1)	1 (2)	18

Tableau 2. Effectifs (phase 1, situation 3) selon l'invariant de la répétition et les algorithmes

Seuls trois binômes (tous de 1e V) écrivent correctement en langage Alpro l'invariant de la répétition à partir de l'algorithme M(R)

A + 0,03 Sto A
A × A + 1

Nous allons examiner maintenant l'ensemble des propositions des binômes.

La catégorie « autres »

Nous regroupons dans cette catégorie :

- Un invariant écrit dans un langage Alpro complété par des mots qui concernent l'amélioration d'Alpro.

Les touches mémoires : Ex. : A, B, C…
Touche pour mémoriser une longue expression sans passer par « = » avant d'appuyer sur les touches mémoires (les étapes d'appuis, non pas le résultat)
Ex. : Le groupe x2 + 1, x ∈ [- 3 ; 2] x1 = x2 + 0.03
Ou le groupe : Ans × Ans + 1

- Deux autres groupes de touches concernent les mémoires variables A, B, C et ne peuvent être identifiés à aucun algorithme de calcul :

- Mémoriser l'expression (Shift STO A ; B ; C) ; Mémoriser l'écart 0,03 ;
- 0.03 = Sto A ; A × A + 1 = Sto B ; B × B + 1 = Sto C ; C × C + 1 = Sto A

La disparition de l'usage exclusif de Ans

Aucun programme n'utilise exclusivement la mémoire Ans (contre quatre binômes dans la situation 2) : l'algorithme M(R)' disparaît.
Les mémoires variables sont donc disponibles pour un plus grand nombre de binômes par rapport à la phase 2 de la situation 2 : 14 contre 12.

Présence forte des mémoires variables

Un autre observable de la présente plus forte des mémoires variables est le nombre plus important des groupes de touches associés à l'algorithme M(F).

La lettre « n » est présente dans deux groupes de touches utilisant l'algorithme M(F) dans l'institution vietnamienne, pour indiquer « algébriquement » la variable compteur :

A × A + 1 ; A = - 3 + 0.03n ; n = 1, 2, 3, 4 …

Dans cette écriture, l'égalité « = », qui est à la fois le signe mathématique pour le résultat d'une opération et une touche d'affectation d'un résultat dans la mémoire Ans, est utilisée pour affecter (improprement) un nombre dans une mémoire : « *A = - 3 + 0.03n* ».
Ce signe « = » est un observable de la résistance du statut de variable mathématique des mémoires variables A, B et C.

Intéressons nous aux quatorze invariant écrits en langage Alpro, associés aux algorithmes M(R), M(F) et M(F)', et qui utilisent des mémoires variables A, B, C avec ou sans la mémoire Ans.

Quel est le statut des mémoires variables utilisées ?
L'écriture de l'invariant de la répétition ne rend pas nécessaire celle de l'initialisation.
Par contre, l'autre opération fondamentale, celle de la mise à jour des variables et caractéristique de l'effaçabilité des mémoires doit être présente.
Le tableau 3 donne les effectifs des binômes selon qu'ils ont mis à jour ou non les variables utilisées dans leur écriture de l'invariant de la répétition, ou corps de boucle.

Opération / Classes	Mise à jour	
	Présente	Absente
1er F	5	4
1er V	3	2

Tableau 3. Répartition des groupes de touches selon la présence de la « mise à jour » d'une variable

Un peu plus de la moitié des binômes (8 sur 15) mettent à jour les variables dans leur corps de boucle (5 pour l'algorithme M(R) et 3 pour l'algorithme M(F)).
Pour les huit binômes, les mémoires variables A, B et C prennent le statut de variable informatique.

Examinons deux groupes de touches prenant en compte ou ne prenant pas en compte la mise à jour des variables.

Prise en compte de la mise à jour	Non prise en compte de la mise à jour
A + 0.03 Sto B B × B + 1 = B Sto A (1er F)	- 3 Sto A A + 0.03 Sto B B × B + 1 à répéter (Calculator) (1er F)

Tableau 4. Deux exemples de corps de boucle avec ou sans mise à jour directe des variables

La mise à jour dans le premier exemple montre la difficulté de la mise en place des variables informatiques.
La manière attendue de mettre à jour la variable A est une mise à jour « directe »

A + 0,03 Sto A

Alors que le binôme passe par l'intermédiaire de la mémoire B pour initialiser « indirectement » la mémoire A :

A + 0,03 Sto B
B Sto

Cette mise à jour se retrouve chez deux binômes de 1er F.

Absence de la variable compteur dans l'écriture en langage Alpro du corps de boucle pour l'algorithme M(F)

Comme nous l'avons déjà dit, le nombre des binômes utilisant l'algorithme M(F) est plus important que dans la situation 2 (phase 2).
Or, dans cet algorithme, l'écriture de l'invariant de la répétition concerne le calcul de l'image d'un nombre « indéterminé » ou « quelconque » de l'intervalle [a ; b] par la fonction f.
Les élèves continuent à déléguer au robot CALCULATOR non pas la répétition mais le calcul des valeurs de l'intervalle : la variable compteur est absente des sept propositions utilisant l'algorithme M(F). Comme lors de la situation 2 (phase 2), elle continue à être prise en charge par la mémoire papier ou par l'usage de la lettre n pour indiquer un rang quelconque (pratique algébrique).

Peu de validation pragmatique sur Alpro

L'examen des fichiers historiques montre que malgré la consigne « *en vous assurant sur Alpro que la répétition fournit bien les nombres attendus* » seulement la moitié des binômes valident leur groupe de touche sur Alpro. L'autre moitié se contente de calculer à la main les premières images par la fonction f, ce qui ne permet pas une validation de l'écriture en langage Alpro de l'invariant de la répétition.

C'est donc l'enseignant qui va devoir recourir à l'exécution des corps de boucle sur Alpro, pour valider ou invalider leur écriture en langage Alpro.

II. Situation 3 – Phase 2 : institutionnalisation de l'invariant de la répétition en langage Alpro

Pour cette validation par exécution sur Alpro publique, l'enseignant choisit des propositions basées sur les deux algorithmes M(R) et M(F). Il doit aboutir à deux groupes de touche écrite sur tableau et le restant durant la phase suivante.

II.1. Dans la classe de 1er F

Résistance de la touche Ans

L'enseignant choisit la proposition la plus courte des propositions figurant sur les fiches.

268. P : […] Ici je commence par celui qui est le plus court donc ça devrait être plus intéressant. Je vois (il commence à écrire au tableau) Ans fois Ans plus 1. Oui oui, venez là s'il vous plaît.

L'enseignant écrit au tableau :

Ans × Ans + 1

L'examen du fichier historique du binôme qui a fait cette proposition a permis de reconstituer leur programme en actes :

- 3 Sto A 0,03 Sto B A + 2B = Ans × Ans + 1 A + 4B = Ans × Ans + 1

Ce binôme utilise donc l'algorithme M(F) mais n'écrit que le groupe de touche Ans × Ans dans leur réponse.
La classe réagit par des rires moqueurs.

269. P : Il ne s'agit pas de rire. Il s'agit de, je vous rappelle, de se demander si, si … ce qu'on veut que CALCULATOR fasse est bien fait par CALCULATOR en imaginant que ce groupe est répété. Ce groupe est répété, est-ce que CALCULATOR a réussi à faire ce qu'on voudrait qu'il fasse. Votre commentaire, s'il vous plaît. Assez fort, s'il vous plaît. Pour que tout le monde entende bien.

270. E0 : La première fois, ça va marcher.

271. P : Oui ça va marcher pour la 1ère fois. J'aimerais pas répéter car vous êtes capable de défendre votre argument…Je ne répète pas.
272. E1 (un autre) : On sait pas si Ans est le résultat de ça ou pas.
273. E2 : Le Ans, il est stocké.

L'élève explique alors publiquement ce qu'il a effectivement tapé sur Alpro (algorithme M(F)) et non pas ce qu'il a écrit. En fait, il n'a écrit que la dernière instruction permettant de calculer f(x) parce que pour lui, cette dernière instruction est identique dans toutes les répétitions.

274. E (du binôme choisi) : Car vous savez, quand vous savez que **A égale à – 3 et B égale à 0.03** et quand vous faites opération, quand vous faites A plus de quelque chose de B, **A plus B et vous faite égale et là, vous avez Ans. Et dans ce cas-là, Ans c'est x.**

Les interactions entre l'enseignant et les élèves, y compris du binôme choisi, vont porter sur les « manques » de la proposition écrite et que l'on va chercher à compléter collectivement. Pour les élèves du binôme choisi, la mémoire Ans est identifier à x : cela revient à plusieurs reprises dans les justifications de ce binôme. L'enseignant cherche à poser le problème de la mise à jour de Ans.

275. P : Mais le B ou il est, là ? Je vois pas le B. Ou il est ?
276. E (du binôme choisi) : En fait **il ne manquait que le** x. Vous prenez, vous prenez, vous faites - 3 et vous faites – 3 fois un nombre de, de 1 jusqu'à 166. Et là, 166 fois 0.03 et **là vous trouvez x**. En fait si vous faites sur la calculatrice **x c'est, Ans c'est x.**
277. E2 : Mais le Ans, il varie chaque fois là.
278. Es : Oui, il varie.
279. P : Oui d'accord, mais comment concrètement CALCULATOR sur Alpro, d'abord **je ne vois pas le signe égal**. Ou est-ce que je mets ? Je mets là. Ou ici ? Donc on va imaginer qu'Alpro, pardon CALCULATOR imprime ça, après le signe égal. C'est bien ça. Il fait ça. Est ce qu'il va obtenir successivement les valeurs qui nous intéressent ?
280. Es : Non.
281. E (du binôme choisi) : Je veux dire qu'**il a des choses avant mais ici c'est seulement le Ans**, ce qu'il fait à chaque fois.
282. P : Ah, il y a des choses avant ? Je mets A plus B ? Et ?
283. E (du binôme choisi) : Oui. Et égale quelque chose et ce quelque chose, c'est Ans. Oui c'est A plus B mais là on peut pas répéter, c'est seulement ce qu'il répète à chaque fois.
284. E1 : A plus B on peut pas répéter car devant B il y a des choses qui varient de 1 à 166. C'est plus rapide chaque fois que chaque fois il fait A plus B et égale car s'il fait Ans fois Ans plus A, c'est forcement plus long.

L'enseignant essaie de rappeler le rôle de la touche « = » dans la mise à jour de la variable Ans :

285. P : Oui mais on peut pas trancher là ? J'aimerais qu'on tranche, qu'on sache. Car ici vous hésitez. Et ici **le Ans, il vient de quel « égale »** ?
286. E1 : C'est A + B.
287. P : Alors A plus B égale.

L'enseignant écrit au tableau :

A + B = || Ans × Ans + 1 =

L'enseignant reprend la conception « *Ans c'est x* » pour tenter de faire comprendre aux élèves du binôme choisi les lacunes de leur proposition initiale. Puis il conclut à un argument d'autorité pour éliminer cette proposition : « *Je voudrais qu'on soit plutôt d'accord pour éliminer ça* ».

288. P : On va imaginer comme ça. Et il appuie sur A + B et égale. Et il va donner quoi ?
289. E (du binôme choisi) : **En fait c'est x. Là, c'est donné x. Ans c'est x.**
290. P : Oui d'accord. Admettons qu'il y a une certaine valeur de x ici. Il va faire x fois x égale et il recommence ici. Ce n'est pas la valeur suivante de x car c'est le résultat de x, donc il risque d'y avoir une répétition sur les valeurs de f de x. Donc c'est-à-dire, c'est les valeurs de f de x qui sera mis au

carré plus x mais pas les valeurs de x. D'accord ? **Je voudrais qu'on soit plutôt d'accord pour éliminer ça.** Alors j'en prends un autre car je peux pas tout prendre. Alors je vois le groupe de touches à répéter là. (il commence à écrire au tableau) A plus B Sto B et voilà je vois ça comme groupe à répéter.

Un problème de succession ?

L'enseignant choisit alors une deuxième proposition M(R).

Étape de préparation
3 Sto A
0.03 Sto C

A + C Sto B
A × A + 1 =
B × B + 1 =
B + C Sto A

L'enseignant commence par rendre publique le contenu de la proposition.

291. P : Voilà, je vois ça, comme groupe à répéter. Chacun le regarde déjà et s'il veut bien juger.
292. E (du binôme choisi) : A c'est – 3.
293. P : Il est vrai que ce qui est précédé de, comme le débat est venu comme tout à l'heure, c'est – 3 Sto A. C'est marqué ici « étape de préparation ». Moins 3 Sto A et 0.03 Sto C. Alors…

L'enseignant écrit au tableau :

- 3 Sto A	A + C Sto B
0.03 Sto C	A × A + 1 =
	B × B + 1 =
	B + C Sto A

L'enseignant exécute « à la main », étape après étape, cette proposition « *en imaginant que* CALCULATOR *appuie sur les touches* ». Se pose alors le problème de la signification de l'instruction « *B + C Sto A alors, c'est quoi cette étape* ».

294. P : Chacun se prononce ah ? (…) Alors ? Moi, ce que je redis j'essaie de suivre ce qui va, **en imaginant que CALCULATOR appuie sur les touches**, ça. Alors – 3 Sto A, 0.03 Sto C. Alors A + C Sto B alors dans B il y – 2.97. Et il fait A fois A + plus 1. Ca fait 9 plus 1, 10, c'est 10 c'est-à-dire la première valeur de la fonction. Et B fois B + 1, c'est – 2.97 fois – 2.97 plus 1, c'est la 2e valeur de la fonction. **Et B + C Sto A alors, c'est quoi cette étape** (il pointe dans la phase « B + C Sto A » au tableau) ? B c'était – 2.97 et C c'était 0.03 et donc c'est – 2.94 dans A. Et c'est écrit dans le groupe à répéter. Donc je reviens, A c'était - 2.94, plus C c'était 0.03 donc c'est – 2.91 que j'ai mis dans B. – 2.94 fois – 2.94 plus 1 et puis – 2.91 fois – 2.91 plus 1, égale B plus C dans A et je recommence. Est-ce que c'est valide ou pas ?

Pour les élèves « *Il y a une faute de succession* ».

295. Es : Non, non.//
296. E3 : **Il y a une faute de succession**.
297. P: Il y a une faute de succession, c'est-à-dire ?
298. E3 : C'est-à-dire s'il répète tout ça il donne 2e fois de //
299. P : Ah, il y a des répétitions. C'est ça ? Des choses qui vont revenir deux fois ?
300. E3 : Non, la 1re fois ça met //, et la 2e fois on part dans B // Ah, non, non, je me suis trompé.
301. P : Alors, je voudrais que chacun se prononce dans sa tête. Est-ce que c'est valide comme groupe à répéter.// Et ma question c'est qu'est-ce qu'on peut faire plus simple avec la même idée ? Alors, je vous propose quelque chose que j'ai l'impression que ça ressemble un petit peu. Je ne me prononce pas encore complètement là-dessus.

L'enseignant laisse alors en suspens le problème « « *Il y a une faute de succession* » pour passer à l'examen d'une autre proposition.

Quelles images obtient-on ?

302. Voilà (il prend une autre fiche et la recopie au tableau), je vois ici, – 3 Sto A, A plus 0.03 Sto B, B fois B plus 1 égale et voilà, je vois ici, à répéter, ce groupe à répéter. Ca paraît plus simple. (il dessine alors un contour autour du groupe à répéter, que nous indiquons en gras). Ce que je vous demande c'est, est-ce que si CALCULATOR fait ça, est-ce qu'il va bien fournir à l'impression, toutes les images, les valeurs successives ?

L'enseignant écrit au tableau :

- 3 Sto A
A + 0.03 Sto B **à répéter**
B × B = 1

Il demande aux élèves de critiquer cette proposition. Le premier commentaire « *Il ne faut pas répéter la première* », à savoir la valeur de départ, valide le corps de boucle.

305. P : […] Bon, j'ai repris mais vous avez le droit de critiquer ah ?
306. E2 **: Il ne faut pas répéter la première.**
307. P : Il faut pas répéter la 1ère ? Mais pourquoi ?
308. E2 : C'est la valeur de départ et ensuite…
309. P : C'est la valeur de départ. Alors toi, tu proposeras de répéter seulement ça ?
310. E2 : Ouais.

La deuxième critique porte sur le fait que l'on obtiendra toujours la même image. L'enseignant répond à cette critique par une exécution « à la main » du groupe de touches. En faisant cela il rencontre un nouveau problème « *Je ne crois pas qu'il a obtenu l'image de – 3 ?* ». Les avis sont partagés, l'enseignant tranche : « *Non. Il n'a pas obtenu l'image de – 3* ».

311. P : Ouais. Est-ce qu'il y a des avis contraires ?
312. E3: Si on met – 3 dans A alors chaque fois qu'on continue ce n'est pas les valeurs successives de x.
313. E4 : **C'est toujours la même image**.
314. P : Attends, je prends, on va imaginer, alors – 3 est stocké dans A et puis on ajoute 0.03 dans A,
315. Es : Non, ça marche pas.
316. P : Donc dans B alors chacun retient dans A il y a – 3 et dans B il y a – 2.97. Puis il fait – 2.97 fois - 2.97 et plus 1 donc il obtient l'image donc de ? De – 2.97. Qui est imprimé. **Je ne crois pas qu'il a obtenu l'image de – 3 ?**
317. E3: Si.
318. E : Non.
319. P : Non. Il n'a pas obtenu l'image de – 3. Mais regardons la suite. Donc, je reviens ici, Sto A, mais qu'est-ce qui est stocké dans A ? Là (il pointe sur B fois B + 1) Car répéter ça veut dire que CALCULATOR arrive ici (il pointe sur le signe =) et puis il revient là (il pointe sur Sto) Donc qu'est ce qu'est stocké dans A ? Ce qui est là ? Alors, est-ce que ça, ça peut rectifier un peu ? On peut améliorer un peu ?

Comment améliorer la proposition par rapport au problème de l'obtention de toutes les images des xi par f ? Il faut résoudre deux problèmes : obtenir l'image par f de –3 et de tous ses successeurs.

320. E5 : Il faut ajouter, à la fin on va mettre, B Sto au A.
321. P : Oh, là
322. E : Non, non.
323. E2 : Il faut, faut déjà, il faut partir avec A égale à - 3..
324. P : Oui, ca veut dire que je supprime ici (il pointe sur Sto A) et je fais ça ? Je fais ça (il fait une ligne de séparation) ? Et ce qui est à répéter c'est comme ça ?
325. E2 : Non, non, il faut changer.
326. P : Ah, il faut changer ?

L'enseignant écrit au tableau :

- 3 Sto A

A + 0.03 Sto B à répéter
B × B = 1

L'enseignant laisse encore en suspens le règlement complet du problème, tout en validant le groupe de touches à répéter : « *On voit qu'il y a un problème mais on sent que la répétition est assurée, non ? La répétition est assurée.* ».

327. P : Mais est-ce que ce groupe à répéter c'est pas mal quand même ? (il pointe la 2e partie après le trait de séparation).
328. E : Oui.
329. P : Oui. Est-ce qu'on retient quand même ? On va retenir ce groupe à répéter. **On voit qu'il y a un problème mais on sent que la répétition est assurée, non** ? **La répétition est assurée.** Je vous propose un autre groupe de touches (il écrit au tableau) A + 0.03 Sto B, B fois B plus 1 égale et B Sto A. Voilà, je vous l'entoure. […]

Une troisième proposition,

L'enseignant recopie au tableau la dernière proposition.

A + 0.03 Sto B
B × B + 1 =
B Sto A

L'enseignant choisit un dernier groupe de touches dont l'initialisation est « - 3 Sto A ». De nouveau il exécute « à la main » le corps de boucle et émet un avis « favorable » sans savoir si cela est convaincant pour les élèves ou non : « *Ca paraît, ça peut tenir bien la route cette affaire comme groupe à répéter là. On voit qu'il y a des petits peu de problème mais on sent qu'il y a une répétition qui est possible là* ».

335. P : Voilà, A plus 0.03 Sto B, B fois B plus 1 égale et B Sto A. Ca va ? Chaque fois qu'on retrouve une nouvelle valeur de B. Alors ? C'est-à-dire quand même qu'il y a quelque chose qui s'est passé avant ah. Ouais. C'est-à-dire – 3 Sto A.
336. E2 : Non, - 3.003 Sto A.
337. P : - 3.003 Sto A. Bon on va imaginer qu'on démarre, on est…on ajoute 0.03 dans B et on fait B fois B plus 1, alors on obtient bien l'image par f du nombre qui était dans B. Et B Sto A, B Sto A c'est-à-dire que ce qui était là passe dans A et on recommence avec 0.03. **Ca paraît, ça peut tenir bien la route cette affaire comme groupe à répéter là. On voit qu'il y a des petits peu de problème mais on sent qu'il y a une répétition qui est possible là.** C'est possible là, non. Ca se ressemble là. Non ? […] On voit qu'il y a plusieurs possibilités. **Je vous propose deux possibilités qui se dégagent de tout ça.** (il commence à écrire au tableau deux groupes de touches). Si vous voulez, j'efface. Voilà, je vous propose une première idée. Je vous donne le premier groupe de touches. J'essaie de rassembler un peu. A fois A plus 1 et A plus 0.03 Sto A. Qui sert là des groupes à répéter. Ouais ? Il y a certainement des choses à faire mais il y a un groupe à répéter. **On a vu 2e idée qui est apparue**. A fois A plus 1, A plus B fois 0.03 Sto A et B plus 1 Sto B. Tout à l'heure on a vu. 0.03 Sto A et B + 1 Sto B. Voilà des candidats, je les rappellerai des candidats à être répétés.

Il s'appuie sur cette dernière proposition pour l'institutionnalisation des deux groupes de touches prévues. : « *Je vous propose deux possibilités qui se dégagent de tout ça.* ». Et il écrit au tableau :

A × A + 1 = A + 0.003 Sto A	A × A + 1 = A + B × 0.03 Sto A B + 1 Sto B

II.2. Dans la classe de 1er V

L'institutionnalisation dans cette classe se fait par un travail coopératif de l'enseignant et des élèves pour améliorer syntaxiquement le groupe de touches. L'enseignant demande aux élèves de toute la classe l'exécution de chaque proposition sur Alpro.

La validation syntaxique par l'exécution sur Alpro d'un corps de boucle M(R)

Le premier groupe choisi est basé sur l'algorithme M(R).

A + 0.03 → A

$A \times A + 1 = AC/On$

359. P : […] Je vais corriger vos réponses pendant environ 10 minutes et puis après. **Vous allumez Alpro s'il vous plaît.** Ca y est ? Vous regardez au tableau. Je vous répète que pour cette question, **vous devez écrire dans votre feuille le groupe de touches que CALCULATOR a à répéter**. Je note au tableau la réponse du 1er groupe (il commence à écrire au tableau en lisant à haute voix ce qu'il écrit). Je note 1 ce message.

L'enseignant écrit au tableau :

1) $A + 0.03 \rightarrow A$
2) $A \times A + 1 = AC/On$

Le premier problème rencontré est le signe « $\rightarrow$ » qui ne correspond à aucune touche d'Alpro. Ce problème est rapidement réglé : on remplace « $\rightarrow$ » par « Sto », puis on élimine AC/ON déclarée inutile.

360. P : Vous voyez, est-ce qu'il peut exécuter ça ?
361. E1 : Non, dans la 1ère ligne.
362. E2 : Car il n'y a pas de flèche.
363. E2 : Non, il faut appuyer Shift Sto.
364. Es : Non, seulement Sto.
365. P : Vous voyez, il n'y a pas la flèche sur Alpro. Doit-on remplacer par quelles touches ?
366. Es : Shift Sto.
367. P (il note au tableau) : Shift Sto mais est-ce qu'il faut Shift ? On n'en a pas besoin sur Alpro (il barre le mot Shift). Il suffit juste Sto. Et comme tout à l'heure on a déjà amélioré CALCULATOR avec la capacité de répéter donc il va répéter ce groupe de touches...
368. E : On n'a pas besoin de AC/On.
369. P : Oui est-ce qu'on en a besoin ?
370. E : Ce n'est pas vraiment nécessaire.
371. P : Ce n'est pas la peine alors (il barre ce mot)

L'enseignant écrit au tableau :

1) A + 0.03 ~~Shift~~ Sto A
2) $A \times A + 1 =$ ~~AC/On~~

L'enseignant exécute les groupes de touches sur Alpro publique, les élèves pouvant en faire autant sur « leur » Alpro. L'exécution des premières répétitions du groupe de touches déclanche un débat sur la question de l'initialisation de la mémoire variable A : « *Mais on a pas encore A* ».

372. P : Donc je vais faisais comme si j'étais CALCULATOR pour exécuter ce message (puis il commence a taper ce groupe de touches sur Alpro). Vous voyez, A plus 0.003 et…
373. Es : **Mais on a pas encore A.**
374. P : Mais ici j'ai d'abord simplement tapé ce qui est écrit dans le message.

Ce qui a été tapé sur Alpro publique :

A + 0,03 Sto A [Erreur. Mémoire non initialisé]

Le message d'erreur renvoyé par Alpro est aussitôt exploité par l'enseignant.

375. P : Donc qu'est-ce qui manque ici ?
376. Es : A, Il n'y a pas encore A.
377. P : Donc on va prendre quoi pour la valeur de A ?
378. Es : - 3.
379. E1 : - 3 plus 0.03.
380. P : On va donner - 3 à quoi ?
381. E2 : Mais il nous demande d'écrire des touches à répéter ?
382. E1 : Ca ça n'est pas à répéter.
383. E : Mais c'est la première chose et cette chose-là n'est pas à répéter.
384. Es : Mais cette première étape n'est pas à répéter.
385. P : La 1ère étape est-elle à répéter ?
386. Es : Non.

387. P : Oui, c'est ça. Ce qui est à répéter c'est quoi ? C'est là. (il souligne le groupe de touche à répéter). Voilà vous me regardez. Je vais exécuter ce groupe de touches (Il tape sur l'ordinateur en lisant à haute voix).

L'enseignant complète le groupe de touches écrit au tableau :

- 3 Sto A
1) A + 0.03 ~~Shift~~ *Sto A* ⇐ à répéter
2) A × A + 1 = ~~*AC/On*~~

À quelle valeur de x correspond l'instruction « A + 0,03 Sto A » ?:

388. P : Ca c'est la valeur de x qui correspond à quelle valeur de x ?
389. E : C'est – 2.97.
390. P : Oui, c'est – 2.97 le tout au carré et y ajoute 1.

Le programme en acte tapé par l'enseignant, reconstitué à partir du fichier historique :

AC/ON - 3 Sto A [- 3 → A] [Stocké] A + 0,03 Sto A [A + 0,03 →A] [Stocké] A × A + 1 = [9,8209]

L'enseignant passe en revue rapidement les propositions de groupes inexécutables sur Alpro.

391. P: Donc, vous voyez, on a de petites modifications à faire, mais l'essentiel, c'est comme ça. Ou un binôme a écrit comme suivant « enregistrer une expression, Shift Sto A, Shift Sto B, Shift Sto C, puis enregistrer l'écart 0.03 ». Est - ce que Calculator peut comprendre « enregistrer une expression » ? Comme je vous ai déjà dit, il faut lui indiquer des touches à répéter et non pas des phrases. Ou un autre exemple « touches de mémoire par exemple A, B, C…, touche de mémoire d'une expression longue, les étapes à appuyer » vous voyez il ne peut pas comprendre

Il conclue en choisissant une fiche contenant un corps de boucle du même type groupe mais sans l'exécuter, en des contentant de le commenter et de le compléter.

392. Ou un autre binôme comme ça (il écrit au tableau en lisant à haute voix ce qui est écrit). A plus 0.03 Sto A, A mutiplie avec A plus 1. Vous voyez que ces deux-là sont semblables mais il faut bien sûr y ajouter – 3 Sto A. Et on répète ce groupe de touche.

L'enseignant recopie au tableau la fiche du binôme choisi, en rajoutant la numérotation et l'initialisation « -3 Sto A ». Ce corps de boucle, attaché à l'algorithme M(R), restera visible dans la suite de la séance :

1) - 3 Sto A
2) A × A + 1 =
3) A + 0.03 Sto A

Le robot CALCULATOR peut-il comprendre le corps de boucle M(F) ?

L'enseignant choisit un corps de boucle basé sur l'algorithme M(F).
En le lisant, il modifie l'algorithme M(F) présent dans le corps de boucle écrit par le binôme « *A = - 3 + 0.03n* » en « *– 3 + (n – 1)0,03* ». Il évite ainsi de poser le problème du décalage d'indice.
Ici, l'enseignant ne recourt plus pour valider le corps de boucle à l'exécution sur Alpro, mais à ce que peut ou ne peut pas comprendre le robot CALCULATOR amélioré : le robot ne comprend pas la lettre « n », « … », « n égale 1, 2, 3, 4 etc ».

393. P : Il y a deux autres binômes qui ont donné le même groupe de touches. Un binôme a par contre donné le groupe suivant. A égale – 3 + 0.03n et n égale 1, 2, 3, 4 etc.
394. Es (rient) : On ne comprend rien.
395. P : Voyons ensemble, **est-ce que le robot peut comprendre ?**
396. Es : Non.
397. P : Vous voyez il y a la lettre « n » qu'il ne comprend pas. Et « n = 1, 2, 3, 4…» c'est les points en suspension qu'il peut pas comprendre bien que vous vouliez exprimer par là une répétition. Donc ici, il y a le 2e groupe de touches qui peut être utilisé (il souligne ce groupe de touche). Maintenant qui peut modifier ça (il indique le dernier groupe de touche écrit au tableau).

L'enseignant a écrit au tableau les deux premiers corps de boucle M(F) en les recopiant à partir des propositions de deux binômes, le dernier corps est l'aboutissement d'un long discours sur l'algorithme M(F) de l'intervention 398 :

Première proposition recopiée d'une fiche	*Deuxième proposition recopiée d'une fiche*	*Troisième proposition écrite par l'enseignant*
A = - 3 + 0.03n n = 1, 2, 3, 4…	x_n = - 3 + (n – 1)0.03 $f(x_n) = x_n \times x_n$	- 3 Sto A 1 Sto B *A ×A + 1* *A + B× 0.03 Sto A ⇐répéter* *B + 1 Sto B*

398. P : Ce groupe (deuxième) s'appuie sur quelle formule pour calculer les images de f ? Non ? Ici il utilise la formule x_n = - 3 plus avec quoi ? – 3 + (n – 1)0.03 (il note cette formule dans un coin au tableau). Par exemple, la 1ère valeur de x, n = 1 donc x est égale – 3 + (1 – 1)0.03 ce qui est égale à – 3. Une fois trouvé x_n, on calcule $f(x_n)$ suivant la formule $f(x_n) = x_n \times x_n$. Est ce qu'il y a un groupe a utilisé cette formule pour calculer f(x) ? Pour écrire un message à CALCULATOR ? **Donc je vais donner un groupe de touche en me basant sur ça**. (il commence à écrire la troisième proposition) D'abord il faut stocker – 3 dans A et puis on prend la valeur de n dans B. Après ça on calcule A × A plus 1 et puis calculer A + B multiplié par 0.03 et stocker dans A. Ca c'est la valeur de x comme vous voyez dans cette formule-là. Et après ça on augmente le rang de x d'une unité. Donc c'est B + 1 dans B. Et le groupe à répéter c'est le groupe là (il souligne au tableau le groupe à répéter).

L'enseignant valide le groupe de touches qu'il vient d'écrire en l'exécutant sur Alpro publique. Ce corps de boucle, attaché à l'algorithme M(F), restera visible dans la suite de la séance :

399. P : Ce groupe de touches est un peu plus compliqué que l'autre précédent. Aucun groupe ne l'a trouvé…Je vais vous aider à exécuter ce message (il le tape sur Alpro publique). D'abord j'attribue - 3 à A puis 1 à B et je fais A × A plus 1. Ca, vous obtenez 10. C'est la valeur de la fonction en quelle variable de x ?
400. Es : En – 3.
401. P : Je continue A + B multiplié par 0.03 et le l'enregistre dans A. Vous voyez Stocké et ensuite que je fais ? Je prends B plus 1 dans B. Alors maintenant si je veux répéter, je fais quoi ? Quel groupe à répéter ?
402. E : A multiplié par A plus 1.
403. P : Vous avez, c'est la valeur de f en ?
404. E : En – 2.97.
405. P : Et puis je répète comme ça.

Le programme en actes tape sur Alpro publique est le suivant :

3 Sto A [- 3 → A] [Stocké] 1 Sto B [1 → B] [Stocké] A × A + 1 = [10] A + B × 0, 03 Sto A [A+B× 0,03-->A] [Stocké] B + 1 Sto B [B + 1 → B] [Stocké] A × A + 1 = [9,8209]

L'enseignant termine cette synthèse en attirant l'attention sur les deux groupes de touches laissés au tableau :

1) - 3 Sto A 2) A × A + 1 = 3) A + 0.03 Sto A	- 3 Sto A 1 Sto B *A ×A + 1* *A + B× 0.03 Sto A ⇐répéter* *B + 1 Sto B*

406. P : Donc vous avez deux groupes à répéter possibles.

En résumé

L'institutionnalisation des groupes de touches s'appuie sur une validation des premières répétitions des groupes de touches proposés par des binômes.

Cette validation est faite par une exécution :

- « à la main » dans la classe 1er F

- sur Alpro publique dans la classe 1er V, l'enseignant jouant le rôle du robot CALCULATOR).

Pour certains binômes, la répétition signifie rester « à l'identique » durant le déroulement de l'algorithme, ce qui justifie (pour eux) la proposition du groupe de touche « Ans × Ans ».

Dans les deux groupes de touches institutionnalisés, la condition d'initialisation est présente (classe de 1e V) ou absente (classe de 1e F).
Par contre, toutes les deux comportent la mise à jour des mémoires variables qui, de ce fait, prennent officiellement le statut de variable informatique dans les corps des boucles institutionnalisés.

Quels programmes informatiques vont être écrits à la machine (Alpro, CALCULATOR II) qui devient une machine ordinateur de Von Neumann ? Quelle proposition de mots pour faire évoluer le langage Alpro ?

III. Situation 3 – Phase 3 : comment « écrire » à CALCULATOR II la répétition ?

Rappelons que l'enjeu de cette phase est de faire évoluer le langage Alpro en un langage (Alpro, CALCULATOR II) qui permette d'écrire un message le plus court possible à la machine. Pour cela, nous avons fait le choix de compléter le langage Alpro par au plus 5 mots que peut comprendre le robot amélioré CALCULATOR II.

Enoncé de la phase 3 de la situation 3

Phase 3 (15 minutes)
Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR II, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.
Comment le lui dire ?
On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.
Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II
CALCULATOR II, en exécutant ce message, doit imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$
Message :
Longueur du message :
Pour calculer la longueur du message, les mots seront comptés comme les touches.
Liste de mots :

Les deux groupes de touches laissés visibles au tableau par les enseignants de deux classes observées à la fin de la phase 2 de la situation 3 sont donnés dans le tableau 5.

Classe de 1er F		Classe de 1er V	
Groupe de touches M(R)	Groupe de touches M(F)	Groupe de touches M(R)	Groupe de touches M(F)
A × A + 1 = A + 0.003 Sto A	A × A + 1 = A + B × 0.03 Sto A B + 1 Sto B	*1) - 3 Sto A* 2) A × A + 1 = 3) A + 0.03 Sto A	- 3 Sto A 1 Sto B *A × A + 1* *A + B× 0.03 Sto A ⇐répéter* *B + 1 Sto B*

Tableau 5. Les corps de boucles laissés visibles dans les deux classes observées

Nous faisons hypothèse que l'écriture des programmes est influencée d'une part par la présence des deux groupes de touches institutionnalisés et d'autre part par les interactions de la phase d'institutionnalisation précédente.

Quels sont les mots que CALCULATOR II doit comprendre, pour que la machine (Alpro, CALCULATOR II) exécute le programme ?

En ce qui concerne l'évolution du langage de programmation, les mots proposés pour la répétition et leur effectif sont donnés dans le tableau 5 ci-après.

	Répéter	Bloc	Refaire retaper	Numéroter
1er F	4	3	2	1
1er V	7	2	1	0

Tableau 5. Éléments d'un langage de programmation évolué

Comme on pouvait s'y attendre (effet de contrat), le mot principal pour exprimer la répétition est le mot « *répéter* » : 11 programmes. Bloc signifie l'usage des signes spatiales pour délimiter la boucle : 5 programmes. Un programme en 1er F utilise la numérotation de ligne pour marquer la répétition (boucle « retourner à »).

Nous donnons la liste des autres mots n'exprimant pas la répétition qui sont proposés par des binômes. Nous indiquons entre parenthèses l'effectif (si supérieur à 1) de chaque mot dans des programmes :

1er F : avec, afficher, après, exécuter, fois (4), la, les, nouvelle, phase, puis, remplacer, secondes, valeurs (2), 1) ; 2) ; 3) ; 4) ; 5)
1er V : avec ; calcul; depuis ; derniers ; groupe de touches ; fois (5) ; message ; suivant ;

A noter que le mot « fois » est introduite par 10 binômes (4 ; 6).

Quel algorithme est programmé ?

Le tableau 6 ci-après donne la répartition des binômes suivant que les programmes produits sont attachés à M(R) ou M(F).

Programmes / Classes	M(R)	M(F)	Pas de réponse	Total
1er F	8	1	1	10
1er V	4	4	0	8
Total	12	5	1	18

Tableau 6. Répartition des binômes selon les algorithmes M(R) ou M(F)

Un seul binôme de 1e F n'a pas donné la réponse. C'est toujours le même.
Comme dans la situation 2, l'algorithme M(R) est prédominant par rapport à l'algorithme M(F) :

- Aucun binôme ne passe de l'algorithme M(R) à l'algorithme M(F) entre les phases 1 et 3 ;
- Cinq binômes passe de l'algorithme M(R) à l'algorithme M(F) entre les phase 1 et 3 ;

La complexité de l'algorithme M(F) et de son écriture en langage Alpro est une raison principale.

Effets de l'institutionnalisation des corps de boucle sur les programmes écrits

Tout d'abord, tous les programmes de la classe de 1e V reprennent l'initialisation des variables présente dans les corps de boucles institutionnalisés dans cette classe, alors que seulement quatre programmes (sur 9) le font en 1e F. L'évocation orale de cette initialisation dans la classe de 1e F n'a pas suffi.

Dix sept binômes (9 en 1er F et 8 en 1er V) écrivent des programmes dont la boucle (invariant de la répétition en langage Alpro) est correcte. Comme on pouvait s'y attendre, ces boucles sont des reprises des corps de boucle, laissés visibles au tableau après la phase 2.

Un seul programme (1er F) conserve le corps de boucle qu'il avait proposé dans la phase 1 de cette situation. L'invariant de la répétition écrit par ce binôme est « *A + 0.03 Sto A ; A × A + 1 Sto B ; B × B + 1 Sto C ; C × C + 1* » avec l'initialisation « *- 3 Sto A* ».
La phase 2 (institutionnalisation des groupes de touches) joue donc un rôle crucial pour l'aboutissement de l'écriture de programmes informatiques comportant un corps de boucle.

Voici deux exemples de programmes produits :

Programme M(R)	Programme M(F)
- 3 Sto A Répéter groupe touches 164 fois A + 0.03 Sto A A × A + 1 = (1er V)	A × A + 1 = [Répéter] A + B × 0.03 [Remplacer] A B + 1 [Remplacer] B (1er F)

Boucle et condition d'arrêt

Dans l'analyse *a priori* nous avons distingué deux types de boucles selon que l'arrêt de la boucle (nombre de répétition) est placé avant ou après la boucle. L'observable de ces programmes est la position des « marques » de la répétition.
Le tableau 7 donne la répartition des programmes (17 programmes sur 18) selon la position de la condition d'arrêt dans le programme.

Programmes / Classes	Avant	Après	Ne pas présent	Total
1er F	3	4	2	9
1er V	3	5	0	8
Total	6	9	2	17

Tableau 7. Répartition des binômes selon la place de la condition d'arrêt dans les programmes

Le nombre des programmes où la condition d'arrêt est placée après, est supérieur au nombre de programmes où la condition d'arrêt est placé avant : 9 contre 6. Ce résultat va dans le sens des résultats d'études précédentes :

> La structure « répète/action/jusqu'à/condition » est la plus accessible (Rogalski 1985, op. cité. chapitre A1)

Plus de la moitié des programmes (5 sur 9) *plaçant la condition d'arrêt après* le corps de boucle utilise un signe spatial (un crochet par exemple) pour séparer l'invariant de la répétition :

- de l'initialisation du programme
- et de la condition d'arrêt.

Nous en donnons un exemple ci-après :

Un programme M(R)
- 3 Sto A $A \times A + 1 =$ $A + 0{,}03\ Sto\ A$ } *phase* Refaire *phase* 166 fois

Les premiers éléments de la notion de « bloc » de programme sont présents dans une telle écriture.

La longueur des programmes (somme du nombre d'appuis de touches d'Alpro et du nombre de mots proposés) est donnée dans le tableau 8 ci-après.

	Programme M(R)	Programme M(F)	Pas de réponse	Ne pas écrire de programme	Total
1er F	[19 ; 32] (7)	22 (1)	1	1	10
1er V	[23 ; 32] (3) ;166 (1)	37 (2), 32 (1)	1	0	8

Tableau 8. La longueur des programmes

Un binôme en 1er V a assimilé la longueur du programme au nombre de répétitions. On peut noter que pour un même groupe de touches institutionnalisé la longueur du programme correspondant peut être différente. Cela vient essentiellement du fait que le nombre de mots proposés est différent selon des binômes.
Deux binômes (un dans chaque classe) a compté le nombre 166 comme un mot au lieu de compter les touches « chiffre » composant ce nombre.

En conclusion

Les programmes ont bien été raccourcis de façon optimale grâce à la capacité attribuée par les élèves à CALCULATOR II de décider de répéter un invariant de calcul par mémorisation d'instructions itératives comme « Répéter ».

Cette phase a donc permis aux élèves de participer à l'évolution du langage Alpro vers un langage comportant des mots, qui organisent les programmes selon deux structures :

- la séquentialité : déjà présente dans le langage Alpro
- l'itération : nouvelle structure du langage destiné à (Alpro, CALCULATOR II)

Les conditions mises en place, écriture d'un message le plus court possible à une machine de Von Neumann pour obtenir l'exécution d'un algorithme itératif de calcul a permis pour la plupart des élèves cette émergence.

IV. Phase 4 : synthèse des programmes écrits et introduction à quelques notions de base d'Informatique

Le temps imparti à la classe 1er F n'a permis à l'enseignant de faire qu'une courte synthèse à partir des affiches des élèves.

Nous n'examinerons ici que ce qui s'est passé dans la classe 1er V.
La phase 4 s'est déroulée dans l'ordre suivant :

- Les programmes écrits sont recopiés sur papier de grand format et rendus publiques par affichage devant toute la classe ;
- L'enseignant passe en revue chacun des programmes et en choisit quatre ;
- Il commente chacun des quatre programmes : longueur et liste des mots proposés ;
- Il conclue en introduisant quelques notions de base d'Informatique en relation avec l'ensemble de l'ingénierie didactique réalisée.

Les quatre affiches choisies, notées M1, M2, M3 et M4, que l'enseignant va examiner publiquement, sont donnés dans le tableau ci-après.

M1 Message : - 3 Sto A Répéter ~~continuellement~~ groupe touches 164 fois A + 0.03 Sto A A × A + 1 = Longueur du message : 23 touches Liste de mots : Répéter ; ~~Continuellement~~ ; Groupe touches ; Fois	**M2** - 3 Sto A 1 Sto B $A \times A + 1 =$ $A + B \times 0.03 \ Sto \ A$ $B + 1 \rightarrow B$ } Répéter encore 164 fois avec A + 0.03 Sto A 1 Sto B Longueur du message : 37 touches Liste de mots : Répéter encore ~~164~~ fois avec
M3 - 3 Sto A 1 Sto B $A \times A + 1 =$ $A + B \times 0.03 \ Sto \ A$ $B + 1 \rightarrow B$ } Répéter ~~pareil~~ 164 fois avec ~~A = x + 0.03 x ∈ [- 3 ; 2]~~ A + 0.03 Sto A 1 Sto B Longueur du message : 37 touches Liste de mots : Répéter ; fois ; avec	**M4** - 3 Sto A A + 0.03 Sto A A × A + 1 Sto B B × B + 1 Sto C C × C + 1 ~~Répéter identiquement précédent avec B, C~~ Répéter encore depuis début Longueur du message : 166 Liste de mots : ~~Répéter identiquement précédent avec B, C~~ Répéter encore depuis début

Tableau 9. Les quatre affiches examinés publiquement par l'enseignant

L'enseignant passe en revue d'abord la première affiche en vérifiant la longueur du programme qui est de 24. Ce programme est la ré-écriture en langage évolué de l'algorithme M(R).

La question « *Mais pourquoi c'est différent de nous ? Nous on en a trouvé 27* », permet à l'enseignant de revenir sur l'enjeu : utiliser le plus petit nombre de mots possible.

Première affiche M1

451. P : Vous regardez au tableau, s'il vous plaît. Vous avez des messages des binômes (suivant l'ordre dans lequel les messages sont collés au tableau). S'il vous plaît. Pour le binôme 1, on doit répéter 164 fois. Quelle est la longueur du message ?
452. Es : 23.
453. P : Vous voyez -, 3, Sto, (il compte publiquement les mots dans le message), ça fait 25.
454. E : Oh, il manque que A × A + 1.
455. P : Vous voyez c'est ça A × A + 1.
456. E : Mais pourquoi c'est différent de nous ? Nous on en a trouvé 27.
457. P : la liste de mots, c'est « Répéter ; groupe touches ; fois » donc ça fait 4

Il examine alors la deuxième et la troisième affiches dont les invariants sont ceux de l'algorithme M(F) écrits en langage Alpro évolué. L'enseignant attire immédiatement l'attention sur le fait que la longueur du programme est plus long en donnant en même temps la raison : « *ici ce binôme s'appuie sur le 2e groupe de touches et ils ont trouvé 37 touches. Ce message est donc plus long.* ».

Deuxième et troisième affiches M2 et M3

458. P : Alors, passons au 2e message, **ici ce binôme s'appuie sur le 2e groupe de touches et ils ont trouvé 37 touches. Ce message est donc plus long**. Et les mots qu'ils ont proposés est « répéter ; encore ; fois ; avec ». Donc 4 mots. Le 3e message est pareil que le 2e. Il y a aussi 37 touches.
459. Es : Ils se sont recopiés alors.

Le problème de la longueur du quatrième programme est soulevé par l'enseignant « *Mais pourquoi la longueur du message est 166 ? Pourquoi ?* » et non refermé.

Quatrième affiche M4

460. P : Le 4e message, les mots qu'ils ont proposés sont « Répéter ; encore ; depuis début ». **Mais pourquoi la longueur du message est 166 ? Pourquoi ?**

Puis l'enseignant rappelle la longueur des premiers programmes : « *environ 2 mille* » et pose une question cruciale : « *Pourquoi on peut le réduire autant ?* » à laquelle plusieurs élèves répondent : « *Le robot CALCULATOR peut maintenant comprendre le vietnamien* ».

461. Les autres messages sont assez semblables que ceux qu'on vient de voir. Je veux vous faire deux remarques. D'abord en ce qui concerne la longueur du message. Tout à l'heure vous l'avez trouvé combien ? Quelques milles n'est ce pas ?
462. Es : 2 milles 3 cents. 2 milles.
463. P : Oui, vous voyez environ 2 milles. N'est-ce pas ? Alors que maintenant on l'a réduit à environ 30 touches. Pourquoi on peut le réduire autant ?
462. Es : **Le robot CALCULATOR peut maintenant comprendre le vietnamien.**

L'enseignant va alors poursuivre sur la réduction considérable de la longueur des nouveaux programmes grâce à la nouvelle capacité du robot CALCULATOR II. Il soulève ensuite deux questions concernant le langage évolué et l'architecture de la machine ordinateur en se référant au couple (Alpro, CALCULATOR II).

À propos du langage

463. P : **Car vous lui attribuez une nouvelle capacité qui est de savoir répéter et la capacité de contrôler la répétition. Donc s'il comprend, il sait répéter et il sait contrôler la répétition alors vous pouvez réduire considérablement le message**. Donc pour cette séance, ce que je voudrais que vous reteniez, ce sont les premières notions d'Informatique. Lorsque vous apprenez la programmation, ce sont aussi les premières notions. Donc pour ceux qui poursuivront en Informatique, en langage Pascal, ou pseudo Pascal qu'est ce qu'on peut écrire ? Alors on donne, pour ce message (affiche M1 et écrit en même temps au tableau) – 3 on donne à quoi ? C'est à A et 1…Et pour répéter en Pascal on utilise le mot « repeat » en anglais. Et A fois A plus 1 puis A + 0.03 Sto A. Et répéter…en Pascal ça veut dire qu'on répète ça 164 fois.

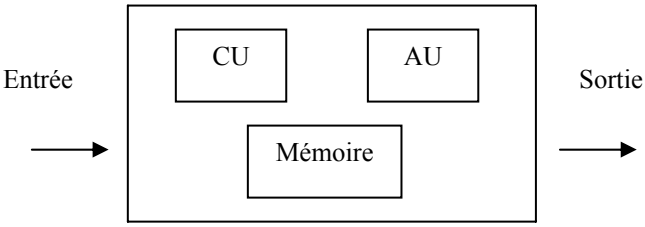
L'enseignant écrit en même temps au tableau :

- 3 Sto A
Repeat 164
A × A + 1 =
A + 0.03 Sto A

À propos de la machine ordinateur

464. P : Et lorsque vous écrivez les premiers programmes c'est comme ça. **Vous vous rendez compte aussi qu'à partir d'un très long programme, on l'a réduit à un si court programme grâce aux amélioration de la capacité du robot, c'est-à-dire l'ordinateur**. Donc par là je veux vous dire **la 2e chose**. Au début lorsque vous calculez l'expression, la calculatrice Alpro contient uniquement une chose pour calculer (il commence à dessiner au tableau). Cette partie est appelée, en langage de l'architecture d'ordinateur, unité arithmétique ; je le note AU, c'est-à-dire « Arithmetic Unit ». Donc dans la question 1, vous vous servez seulement de ça d'Alpro. Et puis dans la suite, vous avez la mémoire, c'est-à-dire les touches mémoires A, B, C, n'est-ce pas ? **Et dernièrement, vous avez ajouté au robot CALCULATOR la capacité de répéter. C'est-à-dire vous lui avez ajouté unité de contrôle pour contrôler la répétition**. Vous comprenez ? Donc vous avez CU, ça veut dire « Contrôle Unit ». Donc ces 3 composants constituent l'essentiel de l'architecture de l'ordinateur : unité arithmétique, je note au passage qu'auparavant on fabriquait des machines qu'on faisait tourner à la main, comme les machines arithmétiques de Pascal, on n'avait que cette partie-là. Et puis la mémoire qui y est ajoutée et avec ça, en même temps on a l'unité de contrôle. Quelque soit la complexité d'une machine, ces trois parties sont fondamentales. Bien sûr il faut ajouter une partie pour faire entrer les données donc c'est l'entrée et ici on peut sortir sur l'écran ou sur l'imprimante donc la sortie.

L'enseignant dessine en même temps au tableau :



465. P : **Donc deux choses que je voudrais que vous reteniez après cette séance, la première concerne la programmation et la deuxième est à propos de l'architecture de l'ordinateur.** J'espère que cela vous sera utile et vous donnera certaines idées.

L'enseignant a donc introduit les trois composants fondamentaux d'une machine ordinateur :
- unité d'arithmétique,
- unité de mémoire
- et unité de contrôle.

Son intervention, centrée sur l'aspect architectural de la machine ordinateur et sur le langage de programmation, a abordé de façon allusive la complexité des algorithmes, à travers la longueur des programmes.

V. Conclusion

Les programmes écrits des binômes à la fin de cette situation pour l'exécution des algorithmes itératifs M(R) et M(F) sur la machine (Alpro, CALCULATOR II) attestent la présence dans la majorité de ces programmes
- d'une part, des notions de variable informatique et de boucle, un observable étant la mise à jour des mémoires variables,
- et d'autre part, d'éléments d'un langage évolué (Alpro, CALCULATOR II), un observable étant des mots et des signes exprimant la séquentialité et la répétition.

Ces résultats valident donc en partie l'une de nos hypothèses selon laquelle l'écriture d'un programme de calcul répétitif en langage proche de la machine est une condition favorable à l'émergence de la notion variable informatique.
Cependant, du fait que la machine est en partie fictive, le recours à la validation pragmatique d'Alpro ainsi que le travail coopératif d'institutionnalisation entre l'enseignant et les élèves ont joué un rôle prépondérant dans l'évolution des notions informatiques.

Ce recours et ce travail coopératif a permis :
- de valider ou d'invalider des programmes proposés par l'exécution des premières itérations d'un corps de boucle sur Alpro ou « à la main » ;
- de rendre publics et d'officialiser des objets informatiques comme « mise à jour » ; « initialisation », « condition d'arrêt » et « corps de la boucle ».

La présence des corps de boucles, laissés au tableau suite aux interactions entre enseignant et élèves a permis de diffuser les découvertes de certains binômes, à d'autres binômes de s'en emparer, d'autant plus facilement que ces propositions répondaient à un problème de ré-écriture informatique.

La situation 3 a permis d'observer la difficulté de la mise en place d'une variable compteur.

Nous allons examiner de façon qualitative le parcours de deux binômes afin de caractériser la genèse de la notion de variable informatique en relation avec la notion de mémoire effaçable ainsi que l'évolution d'un langage évolué qui accompagne celle de la machine.

Chapitre 4

Deux études de cas à travers l'ingénierie didactique

Dans cette partie, nous allons, à l'aide des résultats écrits sur les fiches, des brouillons, des fichiers historiques et des enregistrements vidéo (binôme 1er F) et audio (binôme 1er V), entrer dans une analyse qualitative de la genèse de la notion de variable informatique liée au processus d'instrumentations des touches mémoires variables A, B, C. Cette étude nous permettra en particulier d'observer l'évolution de l'usage des mémoires depuis la situation 1 jusqu'à la situation 3.

I. Un binôme d'élèves dans l'institution française : des mémoires variables aux mémoires effaçables

Ce binôme est observé par Nguyen Chi Thanh (Ob.)

I.1. Situation 1, d'Alpro «machine arithmétique» à Alpro « machine à mémoires variables »

E1 a travaillé sur Alpro pour les calculs 1 et 2. L'examen de son fichier historique montre qu'il a utilisé la procédure Nd (sans mémoire) et obtenu les résultats attendus dès les premières entrées des nombres v_1 et v_2.
E2 a travaillé avec sa calculatrice personnelle (Casio Graph 35+). Lui aussi a écrit les résultats attendus et donné 79 comme nombre d'appuis de touche pour le calcul 2.
On peut déduire que pour les calculs routiniers 1 et 2 la calculatrice fonctionne comme une « machine arithmétique ».

Calcul 3, Phase d'écriture d'un programme en langage Alpro

À propos du calcul 3 (travail en binôme), l'examen du fichier historique a permis de reconstituer les programmes en acte pour le calcul 3.
D'abord ils ont exploré le stockage dans une mémoire variable en choisissant l'entier 9 puis utilisé la mémoire variable A comme une variable mathématique. Le résultat 18 leur sert à valider le stockage de 9 dans la méoire :

C/OFF C/OFF AC/ON 9 Sto A [9 --> A] [Stocké] AC/ON A × 2 = [18] 6 AC/ON

Ils mettent en œuvre ce qu'ils viennent d'apprendre, usage de A et d'autres mémoires variables pour le calcul 3. Dans les premières manipulations, ils n'utilisent pas l'opérateur « × ». Cette écriture est rectifiée suite à l'information renvoyée à l'écran par Alpro.
Ils « linéarisent » la fonction rationnelle y et n'ont donc pas rencontrer le problème de la division nécessitant l'usage de plus de trois mémoires :

1,254 Sto A [1.254-->A] [Stocké] 2A × A A C/OFF C/OFF A × A + 15 A − 5 Sto B [Erreur Synt. Expression non Valide] 2 × A × A × A + 15 × A − 5 Sto B [2 × A × A × A + 15 × A − 5 -->B] [Stocké] B ÷ 32.1 × A − 27.7534 Sto C [B ÷ 32.1 × A − 27.7534 -->C] [Stocké] C × C − 5 × C + 7 = [874.5339988]

Le programme écrit est :

Tape : 1,254 Sto A
2×A×A×A + 15×A – 5 Sto B

B÷32,1×A – 27,7534 Sto C
C×C - 5 ×C + 7 =

Calcul 3, Phase de décodage d'un programme en langage Alpro

Le programme reçu est du type (A, B, Ans, C) (cf. chapitre D1, première partie). Ce binôme l'a exécuté correctement.

Calcul 3, Phase de rectification

Le résultat de l'exécution de leur programme par l'autre binôme est le nombre 874,5339988. Ce résultat est aussi celui que ce binôme avait obtenu dans la phase d'écriture. Cette égalité les a conforté dans la validité de leur programme initial.
Il n'y a pas de rectifications de leur programme initial.

Calcul 3, Phase de synthèses

Dans la phase de synthèse du calcul 3, le programme de calcul exécuté publiquement sur Alpro est du type (A, B, Ans, C) (cf. chapitre D1, deuxième partie).
On peut conclure qu'à la fin de la situation 1, les mémoires variables A, B, C sont disponibles avec le statut de variable mathématique et la mémoire Ans est officiellement introduite.

I.2. Situation 2, la programmation d'un algorithme de tabulation

Situation 2 - Phase 1 : un premier problème de tabulation

Les élèves ont travaillé en binôme dès cette première phase : les effectifs des binômes utilisant les algorithmes M(R) et M(F) pour donner leurs réponses sont respectivement 3 et 7.
Les deux élèves observés s'appuient sur l'algorithme M(F) et la propriété $(- x)^2 = x^2$ pour le calcul de f(- 0,6).
Ils tapent sur la touche « Ans » pour réutiliser le dernier calcul réalisé. Le statut de la mémoire Ans est donc celui d'une variable mathématique.
Voici le programme en acte du calcul des images par la fonction f de la 6e et la 13e valeurs de x.

5 × 0.2 = [1] AC/ON 2 × 2 = [4] Ans + 1 = [Ans + 1] [5] AC/ON 12 × 0,2 = [2,4] AC/ON – 3 + 2,4 = [-0,6] 0,6 × 0,6 = [0,36] Ans + 1 = [Ans + 1] [1,36]

Situation 2 – Phase 2 : écriture d'un programme de calcul répétitif à une machine

Premier programme avec mémoire Ans : statut de variable mathématique
Ce binôme pense tout de suite à utiliser la mémoire Ans pour le calcul des images. Ils envisagent très vite le problème du coût de la répétition de l'entrée du nombre v dans le calcul v × v où v peut être un nombre décimal « *On est obligé de faire ça chaque fois ?* ».

-E1 : Qu'est-ce qui s'est passé ? C'est un peu ça. A ça, à ça…**On est obligé de faire ça chaque fois** ?
-E2 : Comment tu retrouves ça.

E1 suggère l'usage de la mémoire Ans qui est alors utilisé avec le statut variable mathématique afin d'économiser l'entrée d'un nombre sur Alpro.

Ils écrivent au brouillon :

- 3 = Ans
Ans × Ans – 1
- 3 + 0,03 = Ans
Ans × Ans – 1

E2 met en doute ce programme et propose d'utiliser une mémoire variable.

-E2 : Il y a que du Ans ? Tu fais ça ?

Deuxième programme avec mémoire A pour contenir le pas

Aussitôt, E1 lui dicte un autre programme. Ce programme reste inachevé. On peut noter que la mémoire variable A est utilisée pour contenir une constante, le pas p. L'usage de la mémoire reste associé à l'économie du nombre d'appuis de touche.

62. E1 : Je ne sais pas. x fois x plus 1. Et x plus A fois x plus A.
63. E2 : C'est quoi A ?
64. E1 : C'est l'écart. Et Ans fois Ans.
65. E2 : Et Ans car Ans c'est ça.
66. P : Bon Calculator attend votre message.

Ils écrivent au brouillon :

x × x + 1
(x + A) × (x + A) + 1 = ~~Ans~~
(~~Ans) × (Ans~~
x +

Troisième programme mémoires variables A, B : émergence de l'effaçabilité

Le stockage du nombre dans A est alors envisagé, ce qui leur fait rencontrer le problème de la mise à jour de cette mémoire variable : « *A plus 0.03 est mis dans B* », en même temps qu'est pris en compte la répétition « *t'as A, on fait le calcul avec B et on augmente B tout ça et on retourne comme ça* » qui est marqué dans le programme écrit par « … ».
L'opération effective de mise à jour « *A + 0.03 Sto A* » n'est pas encore disponible. Ils ne discutent pas des capacités de CALCULATOR et attribuent implicitement au robot la capacité de répétition.

67. E1 : Et Stocké, stocké.
68. E2 : Bah, non car…
69. E1 : En fait, t'as raison. **On fait – 3 égale**.
70. E2 : Non, **mais A c'est 0.03**.
71. E1 : Non, attends, **A plus 0.03 est mis dans B. Et B fois B moins 1**.
72. E2 : Mais pourquoi moins 1 ? **C'est plus 1**. Tu vois la fonction.
73. E1 : Et après comme on a déjà B, et après ça fait, c'est B… S…T[1]
74. E2 : Mais tu vois moins 3 c'est x. C'est fou ça. A moins 3.
75. E1 : Et tu fais le tour comme ça. **T'as A, on fait le calcul avec B et on augmente B tout ça et on retourne comme ça**. Ca fait le tout et on est déjà intelligent.
76. E2 (il n'a pas l'air d'accord) : Le résultat est comme ça ?
77. E1 : C'est bon comme ça. **T'as pas besoin de recommencer chaque fois**.
78. P : Il vous reste une minute pour que vous écriviez le message. Même si vous commencez votre message. Mettez votre nom s'il vous plaît…Et vous écrivez.
79. E2 : Donc c'est x égale A ?

Ils écrivent au brouillon le programme en langage Alpro:

1) - 3 = A
2) A + 0.03 Sto B
3) B × B + 1 =
4) B + ~~A~~ 0.03 Sto A

~~A × A + 1 =~~ **…**

Les élèves différencient déjà de ce qui relève de l'initialisation de ce qui relève du corps de boucle.

80. P : C'est bon ? Je peux les ramasser, s'il vous plaît ? C'est bon là. Je ramasse quand même.
81. E1 : **Si on fait comme ça, il ne faut pas faire le tour comme ça en passant par - 3…**
82. **E2 : Mais il faut pas passer**. Tu mets juste A égale x.
83. E1 : OK.

[1] Ce binôme prononce toujours S, T, O au lieu de « STO »

Le programme final en langage Alpro recopié dans la fiche

x ~~3~~ = A –3 Sto A
A + 0.03 Sto B
B × B + 1 =
B + 0.03 Sto A

Les nombres – 3 ; -2,94 ; - 2,88 etc. sont stockés successivement dans la mémoire A et – 2,97 ; - 2,91 ; - 2,85 etc. dans B. Ils donnent l'instruction « imprimer » (en langage Alpro « = ») seulement à la suite de l'instruction B × B + 1 : seules les images des nombres espacés d'un pas p (0,06), en commençant par – 2,94, sont calculées.

Comment ce dernier programme est-il validé ?

Pour le valider, le programme en acte est la suivant :

- 3 Sto A A + 0,03 Sto B B × B + 1 = [9,8209] B Sto A A + 0,03 Sto B B × B + 1 = [9,6436]

L'instruction « B + 0.03 Sto A » est modifiée en « B Sto A ». Ils obtiennent alors les images par la fonction f des nombres - 2,97 ; - 2,94 etc. stockés dans B.
Bien que la répétition soit explicitement abordée au cours de l'écriture du programme, ce binôme ne la mentionne pas dans le programme écrit qui se réduit au groupe de touches à répéter, c'est-à-dire le corps de l'itération.
Ils confient donc la tâche de répéter ce corps à CALCULATOR.
L'institutionnalisation des capacités que possède CALCULATOR s'avère primordiale pour l'écriture du programme.

Situation 2 – Institutionnalisation des capacités et des limites du Robot CALCULATOR

Le programme écrit par le binôme invité au tableau est le suivant :

- 3 Sto A
0.03 Sto B
A + B = × Ans + 1
A + B × 2 = × Ans + 1
A + B × 3 = × Ans + 1
A + B × m = × Ans + 1

La simulation des capacités du Robot par l'enseignant sur la calculatrice Alpro publique aboutit à un nouveau problème, celui de la phase 3 :

Ca va vous amener à un nouveau problème. Parce que finalement on s'aperçoit d'une chose que toutes les actions à CALCULATOR sont écrites sous formes de touches. Donc ce qu'on aimerait savoir c'est il y a combien de touches ?

Situation 2 – Phase 3 : coût d'un programme écrit en langage Alpro à la machine (Alpro, CALCULATOR)

E1 propose de calculer les images par f des nombres x_i, à partir de 0 au lieu de -3. Il cherche à réduire le nombre de calculs en utilisant la propriété de la fonction f « *mais on commence en fait si A c'est 0 normalement, par exemple* ».
Puis il explique à E2 comment calculer le nombre d'appuis de touche « *Ah, ouis ? Tu arrondis, tu arrondis. Au plus petit. Ca fait, égale à 166. Tu comptes le nombre de truc tapé et tu fais fois 166* ». Ils transforment donc l'intervalle [-3 ;2] en l'intervalle [0 ;5] en justifiant le résultat 166 : « *A part ça, à part de 0 et après c'est 0 virgule 0 3 et on compte* ». Ils ne décompte pas l'image par f du 1er nombre, c'est-à-dire - 3.

136. E1 : Oublie cette partie-là. **Mais on commence en fait si A c'est 0 normalement**, **par exemple.**
137. E2 : Non, on commence par moins 3. L'intervalle c'est combien ?

138. Ob. C'est de – 3 à 2.
139. E1 : **Ca fait 5 donc c'est comme si tu partais de 0 à 5**.
140. E2 : Il y 5 a l'intervalle ? Et le pas c'est 0 virgule 0 3.
141. E1 : Mais il y a 5. C'est comme si tu partais de 0 à 5 et donc…avec ce qu'on a déjà fait on a juste à faire…**A part ça, à part de 0 et après c'est 0 virgule 0 3 et on compte.**
142. E2 : Tu comptes ça le nombre de fois.
143. E1 : Tu fais 5 divisé par 0.03.
144. E2 : **Mais on a pas un nombre entier**
145. E1 (sur leur bouillon il écrit $\frac{5}{0.03} \approx 166$) **Ah, ouis ? Tu arrondis, tu arrondis. Au plus petit. Ca fait, égale à 166. Tu comptes le nombre de truc tapé et tu fais fois 166 puis**

Puis ils s'appuient sur leur programme écrit dans la fiche pour faire le calcul du nombre d'appuis.

146. E2 : Voilà.1, 2…Vas-y, marque dans la fiche.
147. E1 : 1, 2…14, 15. (il compte le nombre d'appuis de touche en utilisant le 3^{e} programme écrit sur leur brouillon)
148. E2 : Mais non, ça c'est la même chose que là. Ca c'est la redite de ça. (il pointe sur la ligne barrée « A × A + 1 »).
149. E1 : Mais c'est important, c'est à partie de ça qu'on recommence le calcul. Car ici il faut remettre, enfin le x déjà modifié.
150. E2 : Ouais**. Ca fait 21. Multiplié par 166**.
151. E1 : T'est sûr que c'est 21 (il recompte). 1, 2, 3…21, 22.
152. E2 : Non, ah, bon (**E2 fait le calcul 22 × 166 sur sa calculatrice personnelle**), ca fait 3652. Vas-y, tu marques ça.
153. E (du binôme à côté) : Vous avez trouvé combien ?
154. E2 : 3652.

Ils écrivent sur leur fiche :

$$\left.\begin{array}{l} A + 0.03 \;\; Sto \;\; B \\ B \times B + 1 = \\ B + 0.03 \;\; Sto \;\; A \end{array}\right\} 22 \;\; touches$$

$$\frac{5}{0.03} = 166 \rightarrow 22 \times 166 = [3652\ ]$$

Situation 2 – Phase 4 : amélioration du Robot CALCULATOR ?

Ayant délégué la répétition du corps de boucle à CALCULATOR, le binôme ne voit pas comment faire plus court et cherche donc à améliorer Alpro en proposant l'ajout de la touche « x^2 ».
Rappelons les deux derniers programmes écrits par le binôme observé :

Programme en langage Alpro final	**Dernier programme en langage Alpro au brouillon**
x ~~3~~ = A –3 Sto A	1) - 3 = A
A + 0.03 Sto B	2) A + 0.03 Sto B
B × B + 1 =	3) B × B + 1 =
B + 0.03 Sto A	4) B + ~~A~~ 0.03 Sto A
	~~A × A + 1 =~~ **…**

E1 explique à E2 la délimitation du corps de boucle, c'est-à-dire ce qui distingue l'initialisation du corps de la boucle, « *normalement suivant la logique, tu fais ça et tu retombe à 1) n'est-ce pas ?* ». L'instruction « - 3 Sto A » est considérée par E2 comme faisant partie du corps de boucle.

161. E1 : **Je ne vois pas un message plus court que ça.**
162. E2 : Mais si, déjà //.
163. E1 : C'est l'intervalle…

164. E2 : Par exemple, on ne passe qu'une fois, au début, tu vois (il pointe la 1ère phrase dans leur message « - 3 = A »)

E1 s'oppose à E2 : « -3 Sto A » ne fait pas partie du corps de boucle !.
CALCULATOR, qu'il considère probablement comme un autre élève, peut ne pas comprendre « *CALCULATOR il sait pas faire ça* » et il faut « *pourvoir dire ça à CALCULATOR. Sinon il refait la ligne 1 et on refait toujours la même chose* ».

165. E1 : **Non, non ce n'est pas important car ce truc c'est seulement qu'une fois. Tu mets, au début c'est – 3 mais après c'est plus – 3**.
166. E2 : Donc c'est plus 4 alors ?
167. E1 : Pour quoi 4 ? Mais ce n'est pas important car c'est le nombre de fois dont on parle. Ce qui est important c'est qu'on parte de 2) à 4). (il explique en utilisant leur 3e programme écrit sur le brouillon dont les instructions sont numérotées) **Tu ne retombes pas à 1) mais à 2), tu vois ?**
168. E2 : Mais on est déjà à 2).
169. E1 : **Normalement suivant la logique, tu fais ça et tu retombe à 1) n'est ce pas ? Mais ce qui ne faut pas faire**.
170. E2 : Une fois qu'on a fait entrer, fait B plus 0.03 S T O A on fait plus A plus 0. 03 S T O A ?
171. E1 : C'est juste ce que je veux dire, tu comprends. **Une passe par là (l'instruction « - 3 = A ») juste une fois au début.**
172. E2 : **Donc on a passé une fois au début et après non ?**
173. E1 : Ouais.
174. E2 : Donc on a pas besoin de trouver un truc pour //
175. E1 : **Mais CALCULATOR il sait pas faire ça**. C'est nous qui doit faire ça. C'est notre problème.
176. E2 : Donc tu vois on passe pas par là (il point la 1ère instruction dans leur message).
177. E1 : **Ouais mais il faut pourvoir dire ça à CALCULATOR. Sinon il refait la ligne 1 et on refait toujours la même chose**. Qu'est ce que t'as mis dans la fiche ?

Ils se contentent de proposer de l'ajout d'une touche à Alpro : « x^2 »

I.3. Situation 3 : Co-évolution du langage et de la machine (Alpro, CALCULATOR II)

Situation 3 – phase 1 : Formulation des groupes de touches à répéter

Les élèves réalisent que le programme qu'ils ont écrit (situation 2) ne sort que les images des nombres - 2,97 ; - 2,91 etc., c'est-à-dire des nombres espacés d'un pas p = 0,06, et cherchent à rectifier leur programme. Ils complètent les instructions sur B par des instructions sur A en permutant les rôles de A et de B .

199. E2 : A plus 0 virgule 0 3.
200. E1 : Et ça S T O au B.
201. E2 : **Ouis, c'était au B. On avait fait B fois B plus 1, égale et c'était le résultat. Ensuite on avait fait B plus 0 virgule 0 3 S T O au A.**
202. E1 : D'accord.
203. E2 : **Et ensuite on revenait, ça fait A fois A plus 1. Encore le résultat et on faisait A plus 0,03 et S, T, O, au B.**

Ce que E2 écrit au brouillon

A – 3 Sto A ~~A = 3.003~~
A + 0.03 → StoB
B × B + 1 = …
B + 0.03 → StoA
A × A + 1 = …
A + 0.03 → StoB

La double écriture du calcul x × x + 1 est contestée par E1 qui propose un autre groupe de touches. E1 prend en compte des calculs qu'il a essayé sur Alpro et introduit l'instruction « B Sto A » à la place de « B + 0.03 Sto A ».

Cette ultime rectification du programme stabilise une mise à jour « indirecte » de la mémoire variable A, qui prend le statut une variable informatique A par l'intermédiaire de la mémoire variable B. La mise à jour est conçue comme « retransformer ».

204. E1 : Non, non, ça on ne répète pas la répétition car il suffit de changer au fait, à chaque fois. Par exemple donc S, T, O B.
205.
206. E2 : Il faut savoir répéter ça.
207. E1 : Mais après, regarde tu vois ici on répète plus parce que ici c'est A, ici c'est B. Donc en fait on barre ça et puis on repend B ça, B plus machin S, T, O au A ici car B ça va être ça au fait.
208. E2 : Mais B ça va être ça. A plus 0.03.
209. E1 : **Non, B ça va être ça et après on met ce chiffre-là dans B et on reprend en fait ce chiffre-là dans B au fait et comme ça devient A donc lorsqu'on recommence ici A devient A plus 0. 03, plus encore un 0.03 et comme ça, ça fait une répétition. En fait tu vois, on barre ça et d'ici tout de suite on peut retourner et ça marche…Il faut juste…**
210. E2 : Mais il manque encore une étape.
211. E1 : Quoi donc ?
212. E2 : Mais faut savoir répéter ça (il pointe sur deux instructions « A + 0.03 → Sto B ; B × B + 1 = … »
213. E1 : **Non, tu dois répéter ça aussi (il point sur l'instruction « B Sto A ») car ça permet de remettre de chiffre qu'on a mis dans B**.
214. E2 **: Il faut savoir faire A plus 0.03 S T O au B et faire B fois B plus 1 et ensuite retransformer B…**
215. E1 : En A.
216. E2 : En A, et ensuite A plus 0.03 dans B. Alors le groupe de touches à répéter. (il commence à écrire dans la fiche).
217. E1 : Ensuite, on met – 3 dans A comme départ et on fait …
218. E2 : …**B Sto A** et ensuite on repart mais je ne vais pas le mettre car comme ça on va l'embrouiller.

Le groupe de touches à répéter écrit dans la fiche :

- A + 0.03 → Sto B
- B × B + 1 =
- B STO A

A noter que leur groupe de touches ne permet pas de calculer f(-3).

Une explicitation du calcul de nombre d'appuis de touches

L'observateur les questionne sur le nombre 166 « *mais pourquoi vous prenez, vous avez pris 166 au lieu...* ». Leur raison est claire « [valeur] *entière parce que si on prend au-dessus on va sortir de l'intervalle* ». Ils utilisent donc l'intervalle pour déterminer la condition d'arrêt de la boucle. Pour eux le calcul de l'image de -3 par f ne fait pas partie de la répétition.. Cette raison éclaire les réponses des autres binômes qui ont choisi 166 comme le nombre de répétition.

227. E1 : Parce qu'on est parti de – 3 sur l'intervalle.
228. E2 : Ah oui, on a divisé 5 par 0.03.
229. Ob. Mais ça ne donne pas exactement 166.
230. E2 : Oui ça donne 166,666
231. Ob. : **Mais pour quoi vous prenez, vous avez pris 166 au lieu...**
232. E2 : On a pris la valeur…
233. E1 : Petite.
234. E2 : **Entière parce que si on prend au-dessus on va sortir de l'intervalle**.
235. Ob. D'accord.

À la fin de la synthèse de l'enseignant du travail de l'ensemble des binômes, deux groupes de touche restent visibles au tableau :

A × A + 1 = A + 0.03 Sto A	A × A + 1 = A + B × 0.03 Sto A B + 1 Sto B

Situation 3 – phase 3 : comment « écrire » à CALCULATOR II la répétition ?

Des programmes avec des mots

Le binôme observé écrit un groupe de touches qui produit les images des nombres - 2,97 ; - 2,94 ;- 2,91 etc en se référant à l'algorithme M(R). Rappelons qu'ils effectuent une mise à jour « indirecte » de la variable A à l'aide de la variable B et ne calculent pas f(-3).

Le groupe de touches M(R), laissé au tableau, apporte une simplification de la mise à jour des mémoires variables. Le binôme se saisit immédiatement de cette proposition « simplifiée » par rapport à la leur.

Ils font évoluer le langage Alpro en lui ajoutant l'instruction « répéter *n* fois » pour exprimer en même temps la répétition et la condition d'arrêt, l'instruction « En partant de » marquant l'initialisation. Ainsi, l'initialisation du corps de boucle est prise en compte et est distinguée du corps de boucle.

349. E1 : Au boulot.
350. E2 : **Il faut trouver des mots à utiliser ? A répéter ?**
351. E1 : **Répéter 166 fois.**
352. E2 : **Ca fait 3 mots ça ? Et ensuite…**
353. Ob. Non c'est 2 mots. Il y a en fait répéter et fois.
354. E2 : Ah, **ça compte pas les chiffres** ? Donc répéter, mais les lettres ça ne compte pas comme des mots ? A ça compte pas ?
355. E1 : Non, car c'est d'Alpro.
356. E2 : Donc tout ce qui se trouve là (il désigne Alpro) ne compte pas ?
357. Ob. Non, tout ça, ce sont des touches donc on compte comme des touches. A c'est une touche etc.
358. E2 : D'accord. Ca sera donc répéter 166 fois. A plus A, A fois A plus 1.
359. E1 : A plus 0.03 S T O au A et A fois A plus…
360. E2 : Ca c'est quoi alors (il commence à écrire le message).
361. E1 : Mais je crois pas que c'est parfait.
362. E2 (il écrit au brouillon) : **Répéter 166 fois…Il faut mettre, en partant de …Répéter 166 fois**, A fois A plus 1, A plus 0.03 S T O au A. En partant de, 1, 2, 3, 4 (il compte les mots dans le message). Non.
363. E1 : Tu mets des // je sais pas mais.
364. E2 : Mais avec les touches que tu peux trouver. Je pensais aussi, « à partir de l'intervalle » mais il y a pas de touches donc…T'es obligé Est ce que ça marche ? Tu pars de – 3 et tu fais 166 fois et tu vas arriver à, à 2.
365. E1 : Ca marche, ça marche mais…c'est pas propre, tu vois. **Si on doit écrire, combien de fois on répète, c'est pas…il y a des étapes avant où 5 est divisé par 0,03**.
366. E2 : Mais ça, c'est ce que tu rentres dans CALCULATOR.
367. E1 : Oui, d'accord. Mais je dis que ça marche.
368. E2 : Car en faisant des conditions, soit, tu trouves pas des touches, soit ça fait trop de mots…On marque ça ?

Premier programme écrit dans le langage Alpro évolué

Répéter 166 fois A × A + 1 =
A + 0.03 STO A
En partant de - 3 STO A

Ayant terminé l'écriture du programme plus rapidement que d'autres binômes, les deux élèves réagissent à une question de l'observateur remettant en cause l'intérêt de l'instruction « en partant de » : « *Pourquoi vous écrivez en partant de ? Parce que si vous avez déjà écrit – 3 Sto A* », l'observateur justifiant son intervention par : « *Car il faut que votre message soit court aussi* ».

371. E1 : Non, attends un peu. On a tout le temps.
372. E2 : Si tu trouves un autre truc.
373. Ob. : **Pourquoi vous écrivez en partant de ? Parce que si vous avez déjà écrit – 3 STO A**.
374. E1 : **On écrit juste – 3 STO A et on enlève ça, en partant de. Et répéter c'est juste ça à répéter (il point sur les 2 premières instructions) ce n'est pas tu ça donc tu mets tu ça au début et ça c'est après**.
375. E2 : **En fait je vais faire – 3 STO A et répéter…**
376. E1 : Voilà.

377. Ob. : **Car il faut que votre message soit court aussi.**

Le programme final écrit dans le langage Alpro évolué suite à l'intervention de l'observateur

- 3 STO A
répéter 166 fois A × A + 1 =
A + 0.03 STO A

I.4. Conclusion sur le parcours du binôme de 1er F observé, au travers de l'ingénierie didactique

L'écriture d'un programme de calcul répétitif à la machine (Alpro, CALCULATOR) a permis le passage des touches mémoires variables A, B, C vers les mémoires effaçables. La mise à jour, prise en compte comme une « re-transformation » des mémoires variables est disponible dès leur premier programme d'un calcul répétitif. Cependant on peut attester de leur difficulté à formuler un composant essentiel du corps de boucle qui est la mise à jour des variables informatiques. Ils commencent par une mise à jour indirecte, c'est-à-dire s'appuyant sur une mémoire intermédiaire. Cette mise à jour des variables informatiques est un processus long et difficile. Les groupes de touches institutionnalisés par l'enseignant apportent alors une réponse à ce problème et leur permet de rectifier leur programme final.

Les conditions mises en place semblent leur permettre une prise en compte du problème de la répétition et a rendu possible l'émergence rapide d'éléments d'un langage évolué de programmation à la machine (Alpro, CALCULATOR II).

L'observation fine des interactions du binôme peut donner une interprétation possible au calcul erroné du nombre d'appuis de touches (166) donné par de nombreux élèves : l'initialisation de la variable A (– 3 Sto A) ne fait pas partie du corps de boucle (invariant de la répétition) et donc le calcul de f(-3) est hors du décompte du nombre de répétition du calcul des images. Ou encore, le calcul de l'image par f du nombre contenu dans A, c'est-à-dire f(-3) est considéré comme entreprise avant la mise à jour de la variable A par l'élève : le calcul de f(3) ne fait donc pas partie du corps de boucle.

II. Un binôme d'élèves dans l'institution vietnamienne : de la mémoire Ans à la mémoire effaçable

L'étude de cas sur ce binôme, observé par Annie Bessot (Ob.), a déjà été entreprise dans la situation 1 (cf. chapitre D1, troisième partie).

II.1. Situation 1, d'Alpro « machine arithmétique » à Alpro « machine à mémoire Ans »

Rappelons le programme écrit par le binôme pour le calcul 3 :

1,257 = AC/ON
4 × Ans × Ans × Ans – 9 × Ans + 21 = ÷ 12.5 = AC/ON
Ans × Ans – 7 × Ans + 9 =

Comme nous l'avons montré dans l'étude de cas de la situation 1, ces deux élèves du binôme ont instrumenté principalement quatre touches de la calculatrice : « Ans », « = », « AC/ON » et « C/OFF ».
Rappelons l'instrumentation de la mémoire Ans : l'entrée d'un nombre se fait en tapant les chiffres du nombre, suivi des appuis sur les touches « = » [pour le mettre en mémoire] suivi de l'appui sur la touche AC/ON [pour garder le nombre en mémoire, une fois effacé].

Le programme institutionnalisé par l'enseignant à la fin de la situation 1 est donné ci-après.

1,254 → A

Numérateur d'y : A × A × A × 2 + 15 × A – 5 → B
Dénominateur d'y : 32,1 × A – 27,7534 =
B ÷ Ans

Ce binôme a ressenti le besoin d'une seconde mémoire pour résoudre le problème posé par la division : le programme rendu publique utilise les mémoires A et B.
E1 et E2 vont alors essayé apprendre à stocker un nombre dans les mémoires variables. C'est E1 qui l'a découvert et « l'enseigne » ensuite à E2.

270. E1 : J'ai déjà compris mieux plein de chose. **Par exemple, la façon de mémoriser. On sait comment enregistrer un nombre alors**. On appuie Shift, Sto et on enregistre le résultat.
271. E2 : Alors j'essaie de calculer ceci. C'est 12 et A, c'est ça ?
272. E1 : Non, ce n'est pas 12 A. C'est 12 Shift, Sto A. Sto A.
273. E2 : Oui, OK. Donc dans A, il y a un nombre. 36. Maintenant.
274. E1 : 36 Sto B. Oui, c'est fait.
275. E2 : B ÷ A. Egale. 3. Oui c'est bon.
276. E1 : Tu vois, c'est OK. J'ai compris. […]

On peut affirmer que les élèves de ce binôme ont rencontré les mémoires variables A, B, C.

II.2. Situation 2, la programmation d'un algorithme de tabulation

Situation 2 - Phase 1 : un premier problème de tabulation

Dans cette phase, les deux élèves du binôme travaillent séparément, E1 sur sa calculatrice personnelle (Casio Fx 500A), E2 sur Alpro. Ils donnent tous les deux les résultats attendus :

Quel est l'écart choisi entre deux valeurs successives de x ? 0,2
Calculer les images par la fonction f de la sixième, la onzième et la treizième valeur de x.
f(- 2) = 5 f(- 1) = 2 f(- 0,6) = 1,36

E2 comme la majorité des programmes de cette classe (12 programmes sur 18) utilise l'algorithme M(F) pour taper son calcul. Ci-après son programme en actes :

12 × 0,2 = [2,4] – 3 = [Ans - 3] [-0,6] AC/ON . 6 AC/ON 0,6 × 0,6 = [0,36] + 1 = [Ans + 1] [1,36]

Situation 2 – Phase 2 : écriture d'un programme de calcul répétitif à une machine

Calcul du nombre de valeurs, inachevé
Les élèves du binôme interagissent pour déterminer le nombre de valeurs à calculer.
E1 rencontre tout de suite un problème : « ce n'est pas entier » sans préciser ce qui n'est pas entier.
E2 propose une opération pour dénombrer, puis deux résultat à ce dénombrement : 166 et 167.

39. E1 : De –3 à 2 ?
40. E2 : De – 3 à 2, il y a combien ? De – 3 à 2…
41. E1 : Mais ce n'est pas entier.
42. E2 : **De – 3 à 2, la distance est 5, divisé par 0.03**…
43. E1 : Oah.
44. E2 : **166 nombres ?**
45. E1 : On doit tout écrire ?
46. E2 : **167 ?**

Un travail mathématique à la recherche de l'écart entre les images de deux successeurs
L'écart entre deux valeurs de la variable est constant : *quelle est alors l'écart entre les images par f de deux valeurs successives de la variable* ?
C'est ce problème mathématique auquel vont chercher à répondre les élèves, avec une confusion qui revient tout au long des interactions : « *si on ajoute ça (0,03) ça diminue*. ».

Leur recherche va leur donner l'occasion de travailler sur la notion de variation d'une fonction s: « *ce x^2 diminue, après que x diminue 0,03* ». Au brouillon, ils cherchent à calculer l'écart l'image de x+ 0,03 :

$(x + 0{,}03)^2 + 1 = x^2 + 2x \times 0{,}03 + 0{,}03^2 + 1$.

Ce travail s'accompagne d'un stockage dans la mémoire variable A.

66. E2 : Alors, avec cette calculatrice si simple, comment on peut faire un tel calcul ?
67. E1 : On peut mettre 0,03 en facteur. On commence par – 3 et on n'a qu'à soustraire 0.03, plus 0.03, plus 0.03.
68. E2 : **Est ce que, comme ça, chaque fois, ça diminue une fois de 0.03, le tout ?**
69. E1 : Ouais, ouais, **si on ajoute ça ça diminue**.
70. E2 : Alors, si on calcule 0,03 x au carré et plus 1, ça donne combien ?
71. E1 : x
72. E2 : Ouais, c'est ça.
73. E1 : 0.03 x au carré plus 1 c'est combien ? Ah, attends.
74. E2 : Non, je ne fais que des essais. 0.03 au carré + 1. Oui, c'est bon, 1.0009. Alors, maintenant A c'est ça, c'est cette valeur.
75. E1 : Non, ça ne va pas. Ce x doit diminuer alors…
76. E2 : C'est combien. Tu peux lire ton résultat ?
77. E1 : 9.8209.
78. E2 : Plus, plus A. Non ça ne marche pas.
79. E1 : **Attends, maintenant on va voir de combien ça diminue chaque fois, pour chaque fois ?**
80. E2 : x au carré plus 1. x plus 0.03 plus 1. Oh, mon dieu.
81. E1 : On ajoute 0.03, puis 0.03…
82. E2 : Oui, plus 0.03.
83. E1 : Alors comment on fait pour que lui il écrive tous les résultats ? Voilà, - 3 au carré c'est 9 n'est ce pas ?
84. E2 : **C'est-à-dire la valeur du prédécesseur augmente toujours par rapport à la valeur du successeur de 0.03 x plus ceci. Et on a déjà ceci.**
85. E1 : C'est-à-dire, x^2 moins
86. E2 : Stocké, ah, voilà, voilà, on peut le calculer, chaque fois…Le prédécédant est le suivant plus…Mais on ne sait pas.
87. E1 **:** Ecoute, ce x, ce x^2 diminue, après que x diminue de 0.03 (elle fait les calculs au brouillon). C'est diminué. On soustrait 0.03 aussi. On ne sait pas, normalement, comment on fait. Ceci est égale 0.06…Ca on a déjà pris sur les progressions. Chaque fois qu'on diminue, ça donne n multiplié par 0.03, multiplié 0.03. Par exemple, si on double 0.03. n varie de […]

Ce qu'ils ont tapé jusque là :

0,03 × .0 C/OFF C/OFF 0,03 + 1 = [0,0009] Shift Sto A [Stocké] 10 – 9,8209 = [0,1791] + A = [Ans + A] [1,18].

N'ayant pas trouvé un écart fixe entres deux images successives de f, E2 propose de déterminer les valeurs extremums de la fonction. « *maintenant, il suffit de trouver la valeur maximale et la valeur minimale »,* proposition qui n'est pas compris par E1. E1 est conscience de l'enjeu du problème « *car on nous oblige de trouver toutes les valeurs de la fonction* » alors que E1 pense que c'est n'est pas possible.

On peut noter que la compréhension de ce problème de tabulation, rarement présent dans EMS pose aussi de problème à ce binôme.

Ils cherchent donc à calculer la différence des images par f de deux nombres successifs afin de dégager une règle.

Le problème du calcul de l'écart entre deux images successives est conclue par E1 : « *ici ce n'est pas les valeurs de f(x) qui sont de même distance entre l'une et l'autre mais c'est les valeurs de x* ».

89. E1 : Un message et on peut sortir toutes les valeurs ? Il suffit d'un calcul et on peut avoir toutes les valeurs ?

90. E2 : Ouais…Ou on cherche la valeur maximale et la valeur minimale. En tout cas, on ne peut pas trouver toutes les valeurs, 166 valeurs. **De toute façon, on ne peut pas écrire toutes ces 166 valeurs**, n'est-ce pas ? Maintenant, **il suffit de trouver la valeur maximale et la valeur minimale**.
91. E1 : Mais ça sert à quoi de trouver la valeur minimale et maximale ? **Car on nous oblige à trouver toutes les valeurs de la fonction**. **Cette fonction a pour graphique une parabole**, n'est-ce pas ? Cette fonction, n'est ce pas ? On a trouvé quelle formule ?
92. E2 : **Parabole, ça diminue, ça diminue, ça n'augmente jamais.**
93. E1 : Ouais, c'est ça.[…]
108. E1 : Le problème c'est que on ne comprend pas quoi on doit écrire. Attends, ici ce n'est pas les valeurs de f(x) qui sont de même distance entre l'une et l'autre mais c'est les valeurs de x…

Ils retournent au calcul instrumenté par la calculatrice Alpro en envisageant la répétition des calculs « *On fait quoi alors ? On répète ?* » et en recourant à la mémoire Ans déjà présente dans leur premier programme.

109. E2 : **On fait quoi alors ? On répète** ?
110. E1 : **- 3 Ans et encore moins 0,03 et Ans et moins 0,03 Ans, c'est ça** ?
111. E2 : Le problème c'est qu'on n'a pas trouvé une loi et ce n'est pas parce qu'on ne peut pas appuyer sur les touches.
112. E1 : Une loi c'est comme faire. Et moi, **je n'ai même pas encore compris ce que c'est « écrire toutes les images…** ». On appuie sur la touche [=] et ça sort les valeurs, c'est ça ? Ou on appuie successivement sur la touche [=] et la calculatrice donne les valeurs ? On essaie quand même, - 3 et Ans.

Ce qu'ils notent sur leur fiche de réponse :

~~- 3 AC/ON~~
~~Ans - 0.03 × Ans - 0.03~~

Bien que l'idée de répéter un calcul « *On répète ?* » puis de changer de valeurs des nombres x_i soient venue au cours des interactions, l'usage des mémoires variables ou Ans pour stocker les nombres x_i ne leur apporte pas de solution
Ce binôme est donc en échec dans l'écriture d'un programme dans cette phase.

Situation 2 – Institutionnalisation des capacités et des limites du Robot CALCULATOR

Entre les deux séances, l'enseignant a pu recopier les programmes écrits par les binômes sur deux feuilles qu'il distribue lors de la synthèse (cf. chapitre D2, deuxième partie).
L'échec de l'écriture du programme conduit le binôme observé à se saisir de l'un des programmes figurant dans les deux feuilles et présentés publiquement par l'enseignant (cf. chapitre D2, deuxième partie).

L'examen de leur fichier historique montre qu'il s'agit du programme suivant (expérimenté sur Alpro) ré-écriture en langage Alpro (incorrectement) de l'algorithme M(R) :

```
- 3 Sto → A AC/On
A × A + 1 = AC/On
 A + 0.03 Sto → A
 AC/On
 A × A + 1  =  AC/On
 Répéter 164 fois
```

Situation 2 – Phase 3 : coût d'un programme écrit en langage Alpro à la machine (Alpro, CALCULATOR)

Différenciation entre la touche opération « - » et la touche « (-) » signe d'un nombre sur Alpro

Dans la calculatrice Casio de E1, on peut taper « – 3 + 6 » où « - » est à la fois opérateur de soustraction et signe du nombre. Cela est impossible sur Alpro : il faut taper « (-) 3 + 6 ».

Ayant déjà calculé sur Alpro avec des nombres négatifs sur Alpro au début de la situation 2, E2 l'enseigne à E1.

170. E1 : Shift Sto A. Tu dois prendre ce signe-là. Ce signe moins là.
171. E2 : Mais ce signe moins ne peut pas être au début de la phrase.
172. E1 : Alors quel signe de moins. Il faut que tu prennes ce signe. Voilà. Moins 3. On a déjà appuyé sur Ans. Moins 3, tu essaies encore une fois.

Ils comptent le nombre d'appuis de touches du corps de la boucle dans le programme choisi. Ils ne cherchent pas à justifier ou à retrouver le nombre d'itération. E2 dit simplement « *Pour quoi 164 ? Je pense que c'est 166 fois* » (185).

173. E2 : 0.03 Sto A.
174. E1 : Voilà, combien de touches déjà ?
175. E2 : Doucement, attends un peu.
176. E1 : Non tu comptes maintenant, on verra la suite.
177. E2 : Un, deux, trois…Voilà 10 touches. Cette flèche est comptée pour 2 touches.
178. E1 : 10 touches ? D'accord. On continue. A × A + 1 =. Oui A × A + 1 = […]

184. E1 : Voilà, 17 touches, multiplié par 164 fois. Car la suite c'est la même chose. Oui, tu peux faire le calcul : 17 × 164. Ca fait combien ?
185. E2 : **Pour quoi 164 ? Je pense que c'est 166 fois.**
186. E1 : **Oui 164 fois pour trouver tous les résultats**. Oui, tu peux refaire la division pour voir.
187. E2 : 17 multiplié par 164 donne 2788. Donc 2788 appuis de touches.

Leur réponse :

Nombre de fois 2788
Explication : J'ai calculé le nombre d'appuis pour trouver la valeur $f(x_1)$ correspondant à x_1. Puis on multiplie ce nombre par 164 car j'ai trouvé qu'il y a 164 valeurs comme ça et pour chaque valeur on répète le nombre d'appuis comme avec x_1.
Résumé :
Chaque étape a besoin : 17 touches
164 étapes ont besoin : 17 × 164 = 2788 (fois)

Situation 2 – Phase 4 : amélioration du Robot CALCULATOR ?

Tentative d'enregistrement d'un nombre dans la mémoire A

Ce binôme s'appuie sur un programme qu'ils n'ont pas écrit. Dans ce programme, l'opération de mise à jour dans le corps d'une boucle est correctement mise en place. (cf. chapitre D2, deuxième partie). Cependant ce n'est pas pour autant qu'ils se sont approprié la notion de mémoire effaçable. En effet, cette notion est présente pour eux en actes pour entrer successivement des nombres dans une mémoire variable. Mais ils cherchent toujours à comprendre « *pourquoi on doit mémoriser ce résultat* ».
E1 ressent vaguement un lien entre répéter et changes des valeurs « *Si par exemple on répète ceci. Et on peut changer les valeurs* ».

208. E2 : Alors qu'est ce qu'on veut demander ? Une nouvelle touche par exemple.
209. E1 : Par exemple si on veut répéter ce programme. **Si par exemple on répète ceci. Et on peut changer les valeurs.**
210. E2 : **Qu'est ce que tu veux changer ?**
211. E1 : **C'est-à-dire, la première valeur c'est 0.03 et on l'enregistre dans A et on prend la valeur A. Ca va comme ça, non ?**
212. E2 : Au fait, on n'a pas compris l'essentiel.
213. E2 : Je pense que ce qu'on a fait c'est correct.
214. E1 : **Oui, mais le problème est qu'on ne peut pas tout écrire.** Je pense donc qu'il faut écrire // multiplier par le nombre de fois//
215. E2 : Qu'est ce qu'on a déjà fait ?
216. E1 : - 0.3.
217. E2 : Non c'est – 3.
218. E1 : Oui c'est ça. – 3, plus 0.03.

219. E2 : Et quoi encore ?
220. E1 : Shift Sto.
221. E2 : **Pourquoi on doit mémoriser ce résultat ? Ca sert à quoi ?**
222. E1 : Shift Sto A. C'est pour faire A × A + 1.
223. E2 : Ah, c'est la valeur qui la plus petite n'est ce pas ?

Le programme en actes

- 3 + 0,03 Sto A [– 3 + 0,03 -- > A] [Stocké] AC/ON A × A + 1 = [9,8209]

Émergence de la mémoire effaçable comme pouvant contenir successivement plusieurs nombres x_i

Le fait de pour voir accéder au contenu d'une mémoire permet de vérifier à tout moment le nombre contenu dans une mémoire variable et de vérifier la validité d'un calcul.

La proposition de stocker le résultat du calcul « *A × A +1* » dans A déclenche un débat entre les élèves du binôme sur la relation entre le nom d'une variable et les valeurs contenues dans cette variable.

Pour E2 « *c'est toujours A* » car c'est le même nom alors que pour E2 « *c'est quand même un nouveau A* » car contient des nombres différents. L'affectation implicite de E1 « *après on donne cette valeur à A et puis on répète le processus* » marque une étape fondamentale qui voit le passage du statut de mémoire variable mathématique à celui de mémoire effaçable donc de variable informatique.

238. E1 : A × A + 1, shift Sto A encore. **Maintenant on va voir A est égale à quoi** ? Oui tu effaces tout. Pour voir à quoi ça égale. Oui efface tout. Non, fait plus 1. C'est combien ?
239. E2 : 9.8.
240. E1 : C'est bon. **Maintenant tu fais Shift Sto cette valeur**.
241. E2 : Non, **Ca ne pas peut pas être comme ça car c'est trop grand.**
242. E1 : Non, d'après moi je veux que tu enregistres deux valeurs pour comparer la valeur de A de cette fois ci et la valeur A de la dernière fois. **Si c'est un nouveau A on peut poser A** //
243. E2 : **C'est un nouveau A**.
244. E1 : **C'est un nouveau A**. Alors c'est différent.
245. E2 : **Mais, mais c'est encore ce même A car jusqu'à 164 c'est toujours A.**
246. E1 : **Alors mais c'est quand même un nouveau A ?** Tant mieux alors. On va calculer la 1ère valeur. **Après on donne cette valeur à A et puis on répète le processus**. On répète le même processus mais on remplace seulement – 3 par A. […]

E2 finit par reconnaître le nouveau statut de la mémoire variable « *voyons 9.8, donc c'est un autre A* »

Cet accord permet E1 de formuler l'affectation à la mémoire variable « *il faut toujours lui ajouter 0.03. Alors x_2 est A plus 0.03 et puis on continue de faire Shift Sto A* ».

La déclaration « *x_2 est A* » marque une nette instrumentation des mémoires effaçables : plusieurs nombres x_i correspondent à une mémoire effaçable.

251. E2 : Voilà, le 1ère A est – 2.97.
252. E1 : On va enregistrer A. Non je veux dire qu'on calcule seulement x.
253. E2 : Mais ce A change, tu comprends ?
254. E1 : Je veux dire x change…x égale à cela, x égale à ceci. C'est-à-dire.
255. E2 **: Voyons 9.8, donc c'est un autre A.**
256. E1 : Mais ceci est fixe. Il suffit de changer cela. Cette opération ne change pas.
257. E2 : Pour la suivante, on prend A.
258. E1 : Cette opération ne change pas, c'est seulement ceci qui ne change pas **car il faut toujours lui ajouter 0.03. Alors x_2 est A plus 0.03 et puis on continue de faire Shift Sto A. C'est exact?**
259. E2 : Attends, alors ici, comment on continue ?
260. E1 : Ici on continue, ce A, n'est ce pas est A plus 0.03. Puis Shift Sto A et c'est OK comme ça ? Donc ici le A dans A × A est aussi le nouveau A, n'est ce pas ?
261. E2 : Ouis, c'est le même A.
262. E1 : **Donc comme ça on répète. Ce A**…[…]

Bien que la mise à jour d'une variable soit présente d'une manière implicite, ce binôme n'arrive pas à formuler correctement le groupe de touche à répéter.
Ce binôme a déplacé la tache d'améliorer le robot à celle d'améliorer Alpro « *mais que veut dire « améliorer » ? Ajouter donc une touche « carrée » comme ça on doit pas écrire A × A* ».
La mise à jour, opération fondamentale, récemment et implicitement mise en place, est bien présente dans l'explication de leur groupe de touche « *donc je vais simplement écrire que chaque fois qu'on calcule ce A, on remplace dans ceci. Et on garde les autres* ».
On voit aussi que ce binôme cherche en même temps à améliorer Alpro « *il y aune touche qui enregistre la formule* » mais aussi CALCULATOR sur lequel ils se déchargent de la mise à jour d'une variable.
Celui-ci est donc toujours considéré comme ayant des mêmes capacités similaires aux leurs : « *le CALCULATOR ajoute lui même 0.03 pour chaque nouvelle opération* ».

285. E1 : C'est bon, on va écrire comme ça. Après avoir calculé la 1ère valeur de x, non, la valeur première x_1, on va l'enregistrer dans A. Puis calculer A × A plus 1 et puis on propose CALCULATOR de répéter cette expression en remplaçant x – 0.03 par A.
286. E2 : C'est ça. Et il a une touche pour mémoriser la formule. Enregistrer la formule A × A + 1. Mais lorsqu'on enregistre la formule il ne peut pas faire par lui même.
287. Ppu : Est-ce que vous avez fini ? Je vous rappelle que c'est le robot CALCULATOR qu'il faut améliorer et non pas Alpro.
288. E2 : **Mais que veut dire « améliorer » ? Ajouter donc une touche « carrée » comme ça on doit pas écrire A × A.**
289. Ppr : Vous avez noté ?
290. E1 : **Non, pas encore. Donc je vais simplement écrire que chaque fois qu'on calcule ce A, on remplace dans ceci. Et on garde les autres**.
291. E2 : Alors comme je t'ai déjà dit. Il faut avoir une…On garde la même formule. // **Alors tu écris, il y aune touche qui enregistre la formule mais on ne sais pas comment appuyer** (il rit) et **le CALCULATOR ajoute lui même 0,03 pour chaque nouvelle opération**. Enfin je veux dire ajouter lui-même 0. 03 à A.
292. E1 : dans la formule.
293. E2 : Mais quelle formule ? Il faut écrire la formule. C'est A × A + 1.
294. Ppu : Est ce qu'il reste encore des groupes ?
295. E1 : Oui, monsieur.
296. E2 (en lisant leur réponse) : **Ce me plaît pas trop. Ce n'est pas très faisable**.
297. E1 : **Il reste encore beaucoup, en fait, je veux dire le nombre à répéter. Il diminue la moitié. […]**

La réponse d'amélioration écrite

A × A + 1
Il y a une touche pour enregistrer ↗ et il suffit que le robot calcule la valeur de x ∧ et le remplace dans la formule précédente. (enregistre dans A)

II.3. Situation 3 Co-évolution du langage et de la machine (Alpro, CALCULATOR II)

Situation 3 – phase 1 : Formulation des groupes de touches à répéter

La mise à jour est présente et explicité dans l'écriture en langage Alpro du corps de la boucle qu'ils proposent :

A + 0,03 shift sto A
A × A + 1

Quelles interactions leur ont permis d'écrire ce corps de boucle ?
E2 énonce l'affectation qui se fait continuellement lors de la répétition « *A + 0,03 Sto A et ainsi de suite* ».
L'ordre des instructions du groupe de touche qui est le corps d'une boucle est l'objet de leur discussion au cours de la construction de celui-ci.

Alors que E1 propose de répéter d'abord le calcul de l'image du nombre contenu dans A par f « *c'est A × A + 1, n'est ce pas ?* » puis de mettre à jour cette variable. E1, de son côté propose de mettre à jour A puis de calculer A × A + 1 « *Car tu vois c'est A et on n'a qu'à ajouter 0,03 à A* ». Cette proposition, qui ne calcule par f(-3) est finalement accepté par E1.

326. E1 : On doit répéter le groupe de touche A × A plus 1, n'est-ce pas ?
327. E2 : Mais on doit utiliser les parenthèses. On doit reprendre cette ligne comme un sous titre.
328. E1 : Non si on prend ce groupe pour répéter, ça ne va pas.
329. E2 : Ah, oui, c'est exact. **On ne prend que ce qui à répéter. On va commencer à répéter à partir de ça. Sto A**.
330. E1 : Sto A oui, donc **c'est A × A + 1, n'est ce pas** ?
331. E2 : Non, non, on commence à partir de ça. **A + 0.03 Sto A et ainsi de suite**. Tu vois, ce n'est pas là qu'on commence. A = 0,03 Sto.//
332. E1 : Non, parce que d'abord c'est A et on commence à répéter. **Car tu vois c'est A et on n'a qu'à ajouter 0.03 à A.**
333. E1 : **Tu vois, A plus 0.03 et on obtient un nouveau A. On continue de prend le A alors ce sera pas le même A mais un autre A**.
334. E2 : Ce A oui, on obtient le résultat et on fait la multiplication.
335. E1 : Comme ça on trouve la valeur f(x). **Puis on continue de prendre le A c'est-à-dire ce nouveau A et on lui ajoute 0.03.**
336. E2 : Shift Sto A.
337. E1 : Non, ce n'est pas la peine. On enregistre plutôt celui-là. Car ça c'est f(x) on enregistre seulement x. //
338. E2 : Alors comment on continue ? Ajoute 0,03 à A ?
339. E1 : Oui, tu tapes maintenant. Je te le montre. **– 3 + 0,03. Shift Sto** A. C'est OK ? Puis A × A + 1, vas-y.
340. E2 : Oui et c'est [=], n'est ce pas ?
341. E1 : Oui ça sort le résultat, n'est ce pas ? Et tu fais AC/On. Et **maintenant tu prends encore A et tu ajoute 0.03**. A c'est ce qu'on vient d'avoir avec Shift Sto faits tout à l'heure. Mais pour quoi c'est 0 ? Bon on recommence. Ce n'est pas la peine de calculer f tu calcules seulement les x.
342. E2 : **Attends, A + 0,03 et quoi encore ?**
343. E1 : Sto A. […]

Situation 3 – phase 3 : comment « écrire » à CALCULATOR II la répétition ?

Eléments d'un langage évolué de programmation

Ce binôme, pense presque tout de suite au mot « répéter ». E1 propose d'ajouter « continuellement » mais ce mot est finalement barré de leur fiche. Ils proposent «quatrte mots : répéter, groupe, touche, fois.

Pour écrire la condition d'arrêt, ils se rangent à l'avis de la majorité pour décider du le nombre d'appuis de touches (sur lequel qu'ils n'arrivent pas à conclure depuis la situation 2) : alors qu'ils ont trouvé 166 « *regarde, de 2 à – 3, c'est-à-dire 2 plus 3 ça donne 5. Donc la distance est 5 unités et on divise par 0,03* », ils prennent 164 car « *la plupart ont trouvé 164 fois. Donc on les suit* ».

lls terminent l'écriture de leur fiche, en écrivant assez vite un programme informatique d'un calcul répétitif en un langage évolué (Alpro, CALCULATOR II). La longueur de leur programme est de 23 touches (au lieu de 2788) !

Nous transcrivons leur fiche ci-après

Message :
- 3 Sto A
répéter ~~continuellement~~ groupe touche 164 fois
A + 0.03 Sto A
A × A + 1 =
Longueur du message : 23 touches
Liste de mots : répéter, ~~continuellement~~, groupe, touche, fois

Notez que leur message ne permet pas de calculer f(-3) car l'instruction de calcul « A × A + 1 » est avant la mise a jour « A + 0,03 Sto A » alors que suivant leur algorithme utilisé, ces deux instructions doivent dans l'ordre contraire.
Ci-après les interactions qui ont abouti à leur fiche.

417. P : Oui c'est en vietnamien. Vous avez 15 minutes.
418. E2 : Mon dieu.
419. E1 : Vas-y on commence.
420. E2 : Mais si c'est seulement 5 lettres ce sera impossible. Voyons « lap » : l, a, p (le mot « lap » en vietnamien veut dire « répéter »).
421. E (du binôme à côté) : Mais c'est 5 mots.
422. E2 : Non 5 lettres.
423. E1 : Tu rêves ou quoi, c'est 5 mots.
424. E2 : **Mais je dis que si c'était 5 lettres ça serait beaucoup plus difficile. Donc c'est tout simplement le mot « répéter ».**
425. E1 : **Non, c'est répéter continuellement.**
426. E2 : Le message.
427. E1 : Oui, on écrit alors ?
428. E2 : Oui tu prends ce stylo.
429. E1 : Le message pour calculer ça. (elle compte des mots) 1, 2, 3, 4…
430. E2 : T'as déjà combien ?
431. P : Attention s'il vous plaît. Un mot par exemple « répéter » ça donne 1. Il faut que vous notiez aussi la longueur du message et la liste de mot.
432. E1 : Tu vois, si tu ne mets pas 164 fois, il va faire, faire et il va dépasser 2 // On a déjà trouvé le nombre de fois à répéter mais j'ai oublié la méthode. On va refaire ça, vas-y.
433. E2 : Non, je sais pas.
434. E1 : C'est toi qui l'as trouvé. Alors dis le moi.
435. E2 : **Regarde, de 2 à – 3, c'est-à-dire 2 plus 3 ça donne 5. Donc la distance est 5 unités et on divise par 0,03**.
436. E1 : **Alors ça donne 166 fois.**
437. E2 : **Mais la plupart ont trouvé 164 fois. Donc on le suit.**
438. E1 : Oui.
439. P : Vous avez encore 10 minutes.
440. E2 : 10 minutes encore, que fait-on ?

II.4. Conclusion sur le parcours du binôme de 1er V observé au travers de l'ingénierie didactique

Seul l'usage de la mémoire Ans qui est essentiellement instrumentée par ce binôme dans la phase 1 ne leur a pas permis d'écrire un programme de calcul répétitif. Ils n'ont pas pu non plus reformuler le corps de la boucle. Le calcul du nombre d'appuis de touches leur a permis de prendre conscience de la nécessité de la répétition. La reformulation de propositions au robot CALCULATOR a déclanché des réflexions sur ce qui peut être enregistré dans la mémoire.
Le fait de devoir répéter l'invariant opératoire A × A + 1 =, composant du corps de la boucle (l'autre étant la mise à jour), dans une des mémoires variables a été crucial à l'émergence de la notion de mémoire effaçable, observable intrinsèquement liée à la notion de variable informatique.
Le processus d'instrumentation des touches mémoires effaçables confirme la validité de notre hypothèse de recherche : l'écriture d'un programme de calcul répétitif en langage proche du fonctionnement de la machine est une condition favorable à l'émergence de la notion de variable comme mémoire effaçable.
Durant l'écriture du programme, on peut attester d'un travail mathématique sur des notions comme image d'un nombre par une fonction, variation d'une fonction, graphique d'une fonction. Ce qui confirme notre hypothèse : la formulation de la condition d'arrêt et de l'invariant de la répétition, nécessaire à l'écriture d'un message à une machine à mémoire effaçable, oblige à un travail réflexif sur les objets mathématiques présents dans la solution mathématique d'un problème.

Conclusion et perspectives

I. Les principaux résultats de la thèse

Les notions de boucle et de variable se construisent en même temps que l'architecture de la machine se transforme. Tel est le fil conducteur de l'ingénierie didactique basée sur les résultats d'analyses institutionnelles et épistémologiques et que nous pouvons intituler « d'Alpro machine arithmétique à Alpro unité de calcul dans une machine de Von Neumann ».

I.1. L'effaçabilité de la mémoire et la notion de variable

La notion de variable informatique ne prend son sens qu'à partir du moment où une mémoire est conçue comme une mémoire effaçable, au delà de son rôle de conserver une donnée (ici un nombre). Cette affirmation se nourrit à la fois de l'analyse que nous avons faite de la genèse historique de la machine ordinateur et de la genèse expérimentale dans deux institutions d'enseignement à partir de situations de l'ingénierie didactique. Mais cette notion de variable pour exister a besoin d'une matérialisation en tant que mémoire d'une machine, c'est-à-dire un emplacement réservé à l'avance pour conserver une donnée.

Le statut prédominant des mémoires, à la fin de la situation 1 de l'ingénierie est celui de variable mathématique : une mémoire contient un et un seul nombre durant tout le calcul. Une mémoire n'est donc pas encore effaçable et n'est donc pas une variable informatique. L'exécution sur Alpro du programme écrit par autrui ainsi que les premières rencontres avec les mémoires dans les phases exploratoires des calculs routiniers semblent avoir contribuer de façon cruciale au processus d'instrumentalisation des touches mémoires d'Alpro.

Le stockage des nombres dans les mémoires devient *intentionnel* seulement à la fin de la situation 1 de l'ingénierie. C'est le résultat le plus probant concernant l'instrumentalisation des touches mémoires à la fin de la situation 1. De ce fait la touche Ans, mémoire la plus utilisée, devient une mémoire visible.

Il faut attendre la fin de l'ingénierie pour que la majorité des programmes finaux et écrits des binômes pour l'exécution d'algorithmes itératifs sur la machine (Alpro, CALCULATOR II) atteste de la présence de l'effaçabilité intentionnelle des mémoires par l'opération de mise à jour. Ces mémoires ont alors acquis le statut de variable informatique.

I.2. Le fonctionnement de la calculatrice comme une machine arithmétique

Les calculatrices à la disposition des élèves dans EMS sont toutes munies de touches mémoires associées à des mémoires effaçables. Or l'analyse institutionnelles et l'analyse *a posteriori* de la situation 1 de l'ingénierie mettent en lumière un fonctionnement largement majoritaire de la calculatrice comme une machine arithmétique : les touches mémoires peuvent être utilisé sans donner intentionnellement à la mémoire le caractère d'effaçabilité.

C'est un résultat de l'analyse épistémologique d'avoir distingué machine arithmétique de machine à mémoire effaçable. L'invention de cette dernière a permis l'écriture du premier programme informatique (Adda 1842).

Les calculatrices qui sont entre les mains des élèves de EMS disposent principalement de deux types de mémoires : Ans mémoire du dernier résultat et les mémoires variables, ces dernières étant les seules candidates à permettre la conceptualisation de la notion de variable.

Déjà repéré par les analyses institutionnelles, l'usage de la mémoire Ans a été précisé par l'analyse *a posteriori* de la première situation de l'ingénierie qui a dégagé deux caractéristiques propres à *l'institution française* à propos de calculs instrumentés[1] :

- Une règle du contrat didactique du calcul instrumenté : l'élève est autorisé à prendre comme résultat du calcul la valeur approchée, arrondie ou tronquée, du nombre affiché à l'écran de la calculatrice.
- Une pratique privée (Ans~~Sto~~) de découpage du calcul en sous calculs, découpage permis par le stockage *non intentionnel* du dernier résultat dans la mémoire Ans.

Nous pouvons même avancer que la genèse instrumentale de la calculatrice en tant que machine arithmétique, que ce soit en France ou au Viêt-nam, est inachevée. En effet, l'instrumentation des touches mémoires variables A, B, C et Ans, dépend de celle d'un complexe de touches, comme AC/ON, C/OFF, « = », Sto, Rcl, dont toutes les fonctionnalités sont loin d'être maîtrisées.

1.3. L'économie de la communication homme - machine et la notion de boucle

La présence de calculs répétitifs qui installent dans EMS les algorithmes itératifs a été l'une des justifications de notre intérêt pour la notion de boucle. L'autre a été l'apparition de boucles dans l'écriture du premier programme à une machine à mémoire effaçable (machine analytique de Babbage) pour l'exécution d'un algorithme itératif par Adda (1842).

Nous avons donc fait l'hypothèse que l'intention d'exécuter des calculs répétitifs avec un invariant opératoire est une nécessité pour concevoir et élaborer une communication à la machine qui économise la répétition de tous les calculs exécutés par elle. La notion de boucle est une réponse à cette recherche d'économie.

C'est cette hypothèse qui fonde le passage de la situation 2 à la situation 3, du robot CALCULATOR au robot amélioré CALCULATOR II, qui enregistre le programme communiqué. De ce fait la machine (Alpro, CALCULATOR II) est une machine à programme enregistré, donc une machine de Von Neumann.

C'est aussi un résultat de l'analyse épistémologique d'avoir distingué machine à programme enregistré (machine de Von Neumann) de machine à mémoire effaçable (machine analytique de Babbage).

I.4. L'évolution du langage Alpro vers un langage à une machine de Von Neumann

La machine de Von Neumann, intégrant un programme compilateur enregistré autorise la traduction de tout langage sémantiquement proche du langage naturel en un langage exécutable directement par la machine. C'est ce qui a permis les évolutions des langages depuis le langage machine vers les langages les plus évolués.

[1] C'est dans la situation 1 que l'analyse comparative entre la France et le Viêt-nam a mis en évidence des pratiques instrumentées différentes entre les deux pays (à l'issue du collège). Pour les situations 2 et 3, on peut considérer que l'état du système partage une même ignorance vis-à-vis des objets élémentaires, variable et boucle.

En même temps que l'émergence de la notion de boucle, nous avons conçu le passage de la situation 2 à la situation 3, (de la programmation d'un algorithme itératif à la machine à mémoire effaçable à celle du même algorithme à la machine de Von Neumann) comme nécessitant la participation des élèves à l'évolution d'un langage : aux instructions de séquentialité, présentes dès la situation 1, s'ajoutent des instructions d'itération et de sortie de boucle.
Notamment, la mise à jour des variables par une notation d'affectation en langage Alpro, étroitement associée au processus d'instrumentation des touches mémoires effaçables A, B, C, doit s'articuler à leur initialisation, dans l'économie de l'écriture de la répétition.

I.5. Le travail mathématique nécessaire à la programmation d'un algorithme

L'analyse épistémologique a montré qu'Ada a dû transformer un algorithme mathématique infini en un algorithme itératif fini pour pouvoir écrire un programme à la machine analytique de Babbage.

Les élèves (étude de cas) se sont, eux aussi, engagés dans un travail sur l'algorithme de tabulation pour établir et écrire, en langage Alpro et en langage plus évolué, l'invariant et la condition d'arrêt de la répétition. Ces écritures ont nécessité un retour réflexif sur les objets mathématiques présents dans le problème de tabulation : variation de la fonction, discrétisation de l'intervalle de définition

I.6. Un nouveau type de tâche

L'ingénierie didactique a introduit un nouveau type de tâche informatique « Écrire un programme informatique pour produire des résultats effectifs à partir d'un algorithme itératif (connu comme solution d'un problème mathématique) » dont notre analyse institutionnelle a montré l'absence dans EMS.

L'analyse *a posteriori* atteste qu'il est possible d'en faire la dévolution aux élèves et de mettre en place un travail coopératif d'institutionnalisation de l'écriture en langage évolué d'objets informatiques élémentaires (comme boucle, variable, opérations sur les variables), travail nourri par une validation syntaxique.

1.7. La double nature des machines de l'ingénierie didactique

L'absence institutionnelle de tout langage de programmation au niveau secondaire (aussi bien en France qu'au Viêt-nam) et la possibilité de choisir entre les deux stratégies d'enseignement mises en évidence dans l'analyse des traités, nous a fait prendre la décision de nous appuyer sur le recours à des machines dans l'ingénierie (stratégie d'enseignement du traité de Knuth, 1968).

Les machines présentes dans l'ingénierie didactique ont une double nature : matérielle pour la calculatrice Alpro et fictive pour le robot CALCULATOR.
Or, l'évolution du robot joue un rôle crucial : d'unité de commande ne pouvant pas répéter un calcul dans la machine (Alpro, CALCULATOR) il devient unité de commande pouvant enregistrer un programme (compilateur) dans une machine de Von Neumann (Alpro, CALCULATOR II).

Cette double nature des machines choisies permet un jeu de la validation syntaxique : validation pragmatique par l'exécution des programmes de calcul sur Alpro (situation 1), validation conceptuelle selon des critères de validités avec les programmes à boucle sur la machine de Von Neumann. Ces critères de validités résultent d'un travail coopératif (entre les

élèves et entre les élèves et l'enseignant) d'écriture d'un programme dans un langage évolué (situations 2 et 3 de l'ingénierie).

II. Limites de l'étude et nouvelles directions de recherche

1. L'ingénierie s'arrête au moment où les objets informatiques élémentaires, variable informatique et boucle, se mettent en place. Il n'y a donc pas eu de stabilisation des connaissances sur ces objets dans des types de tâches qu'il reste à inventer.

Un seul domaine de valeur pour les variables (ou type de variable) est envisagé, celui des nombres décimaux. Or d'autres structures de données existent dans EMS comme des tableaux ou des listes.

Comment prolonger notre ingénierie pour donner un sens « informatique » à ces structures de données élémentaires ?

Cette ingénierie favorise la conception d'un type de boucle dont le nombre d'itérations est déterminé à l'avance, ce qui fixe la nature de la condition d'arrêt : boucle « répéter n fois… ». Or d'autres types de boucles existent pour lesquels la condition d'arrêt s'exprime par une condition logique comme : « tant que…faire » ou « répéter…jusqu'à…». Notre choix de privilégier la première boucle a été contraint par la nature d'Alpro.
La restriction des boucles à une seule boucle limite la signification de cette notion informatique.
Le choix d'un algorithme itératif a limité la signification d'une autre notion informatique intimement liée à la sortie de boule : celle du branchement conditionnel.

Doit-on améliorer le robot fictif Calculator pour qu'il enregistre des programmes avec certaines conditions logiques ?
Ou ne suffit-il pas de modifier Alpro par des touches « logique » comme une relation d'ordre « > » (ou plusieurs) pour ouvrir le champ des possibles dans l'écriture de l'itération ?
Ce sont des pistes que nous ouvrons pour prolonger ou modifier l'ingénierie didactique.

2. L'émulateur de calculatrice Alpro a d'autres limites : par exemple, la touche « = » recouvre plusieurs significations comme dans un calcul algébrique ou numérique. L'usage de la calculatrice lui en rajoute une autre : celle d'affecter automatiquement un résultat dans la mémoire Ans.
Dans l'ingénierie, nous imposons une signification supplémentaire, celle d'être une instruction d'impression d'un résultat choisi.
On peut penser que ce choix de l'ingénierie a pu être la source de certaines difficultés concernant la mise à jour des mémoires.
On pourrait pallier assez facilement cet inconvénient en ajoutant là Alpro une fonctionnalité d'impression commandée par une touche « imprimer ».

3. Nous avons fait le choix du problème de tabulation, problème présent dans EMS, pour faire traiter par les élèves un problème de programmation absent de EMS. Or ce problème est pris en charge de façon pertinente par le tableur dans EMS en France.
Certains élèves de 1^{e} S ont d'ailleurs proposé, lors d'une pré-expérimentation de résoudre le problème de tabulation avec le tableur de leur calculatrice.

N'est-il pas possible d'envisager une reprise de l'ingénierie didactique avec Alpro (peut-être amélioré) sur un problème mathématique du domaine de l'arithmétique ou de la statistique pour lequel il n'existe pas de logiciel prenant en charge la répétition ?

4. Si nous avons dans les études de cas conduit une étude de la genèse instrumentale des touches d'Alpro, nous avons laissé de côté celle d'Alpro en tant qu'unité de calcul dans deux machines, une machine à mémoires effaçables (Alpro, CALCULATOR) et une machine de Von Neumann (Alpro, CALCULATOR II).

Quelles conditions, sur la machine et dans une situation didactique, mettre en place pour conduire une telle étude ?

5. Nous rejoignons Ravel (2003) dans le constat de la difficulté à faire vivre l'algorithmique et la programmation dans EMS. Elle déclare à propos de l'arithmétique en terminale S :

> La difficulté à faire vivre l'aspect algorithmique de l'arithmétique par le biais de l'utilisation des calculatrices, de tableur ou d'autres moyens informatiques se retrouve dans les choix de cours des enseignants. Tout comme les manuels, ils n'utilisent que très peu les outils informatiques pour effectuer un véritable travail sur les démarches algorithmiques en arithmétique. (p. 266)

Un autre indice de cette difficulté est, pour nous, la façon dont un enseignant de lycée, ayant participé à notre expérimentation, cherche à contrôler l'usage de la calculatrice. Ce contrôle est exercé par la pratique d'un calcul mental basé sur une estimation du résultat intermédiaire et du résultat final.
Nous illustrons ce contrôle par un extrait de la phase de synthèse du calcul 3.

> **21.** P : Voilà, est-ce que vous pouvez nous faire un petit essai là ? **Je vous laisse faire à la main, vous sortez un petit brouillon et puis vous, rapidement sans faire le, sans poser les opérations. Juste avoir, d'essayer d'avoir une petite estimation de l'ordre de grandeur.** De tête, essayez, l'ordre de grandeur d'y déjà. Et ensuite, l'ordre de grandeur de z. En deux temps. **Un virgule quelque chose, il faut, il faut le remplacer par quoi ?**
> 22. Es : **Par 1.**
> 23. P : 1, ouais, pourquoi pas, ouais. Tu dis combien ?
> 24. E : //
> 25. P : **Environ 4 ?** Tu peux nous donner la fraction que tu as obtenue ?
> 26. Es://
> 27. P : **Environ 16 par 4 ?**
> 28. Es : Ouais

L'enseignant écrit en même temps au tableau :

$$y \approx \frac{16}{4} \approx 4$$

> 29. P : **Environ 4. On fait une grosse erreur si on remplace 1,257 par 1 ?**
> 30. E : Oui
> 31. Es : Non
> **32.** P : **Est ce qu'il y a des basculements de signe ? Il y a un risque de basculement de signe ou quelque chose comme ça ?** Si c'était 1,3, par exemple. 9 fois 1,3, ça fait combien ça ? 9 fois 1,3, un peu près ? Ca fait un peu près moins 11, enfin, moins 9 fois 1,3, ça fait moins 11, moins 12, donc le dessus va rester positif. Ca ne fait pas un changement de signe, ah ? **Et en dessous, est-ce que ça change fondamentalement les choses ?**

C'est donc la pratique régulière d'un calcul mental qui construit un critère de validité (bien fragile) pour la pratique avec la machine arithmétique, qu'est la calculatrice : on perçoit là le bouleversement des praxéologies qu'une remise en cause de « cette idéologie dominante du calcul non instrumenté » doit construire.

Un contrôle algorithmique des calculs appelle d'autres techniques et d'autres théories (à inventer dans EMS ?) que celles proposées par l'enseignant, qui n'est rien d'autre que la reprise, pour la machine, de contrôles arithmético-algébriques d'avant la calculatrice.

Comment construire de nouvelles praxéologies pour ce nouveau type de tâche de calcul instrumenté ? Peuvent-elles s'appuyer sur les praxéologies présentes ?

Notre ingénierie didactique soulève donc un ensemble de questions qui sont, pour nous, au cœur d'une formation des enseignants de mathématiques sur les objets élémentaires d'informatique en relation avec des problèmes mathématiques et en écho avec les fortes recommandations noosphériennes des deux pays pour l'introduction de ces objets, pour l'utilisation de la calculatrice et de l'ordinateur.

Développer notre ingénierie didactique, en incluant l'amélioration de l'émulateur Alpro, pour l'intégrer dans une telle formation est l'une des directions à la poursuite de cette rechercheque nous privilégions.

Tóm tắt luận án

Đối với các nền giáo dục trên thế giới, trong đó có Pháp và Việt nam hai nước liên quan đến công trình nghiên cứu này, từ hai mươi năm nay việc đưa các yếu tố Tin học vào trong việc dạy học môn Toán ở phổ thông diễn ra dưới nhiều cách thức khác nhau.

Các lí do đưa ra bởi các nhà cổ xuý cho việc dẫn nhập các yếu tố này tập trung nhấn mạnh dưới hai khía cạnh sau :
- Do tính chất cơ bản và tính chất phổ dụng của các yếu tố này : hai tính chất này kết nối với nhiều chuyên ngành khoa học như: lôgic, lý thuyết thuật toán và Tin học ;
- Do lợi ích mà việc dạy học các yếu tố này có thể mang lại như việc thông hiểu các công nghệ Tin học đồng thời trong việc biến đổi một vài khái niệm Toán học.

Vượt lên cả sự thống nhất về các yếu tố trên, ta có thể xác định hai hình thức chính sau đây (tồn tại ít nhất là ở Pháp và ở Việt nam) :

- đưa các yếu tố Tin học vào môn Tin học với tư cách là môn học độc lập ;
- đưa các yếu tố Tin học vào môn Toán học ;

Mới đây tại Pháp, uỷ ban Kahane nghiên cứu về giảng dạy môn toán, đã đề nghị « *dẫn nhập một phần của bộ môn Tin học trong môn Toán học và trong việc đào tạo giáo viên* » trong khi phát triển dần dần các nội dung dạy học « *để tích hợp các đối tượng mới và các khái niệm thuật toán và lập trình* ». (Báo cáo của uỷ ban nghiên cứu Kahane 2001)

Trong khi khẳng định sự cần thiết của việc truyền bá sự sử dụng các công cụ Tin học trong việc dạy học toán, đề nghị này là sự nối tiếp của các sự lựa chọn của các nhà trí quyển hiện nay cũng như trong quá khứ. Tuy nhiên, khác với chúng, sự đề nghị này nhằm đến việc nhấn mạnh hơn nữa việc xây dựng các mối liên hệ lí thuyết gắn bó Tin học với Toán học.

Việc so sánh báo cáo này với một báo cáo của uỷ ban nghiên cứu giảng dạy trước đó cho phép ta hiểu thêm sự khác biệt này :

> Chúng tôi không nghĩ rằng Tin học phải được giảng dạy như một môn (lí thuyết) riêng biệt ở cấp bậc phổ thông. Thật vậy, Tin học dạy ở trình độ này sẽ chưa các nguy cơ liên quan đến sự hình thức hoá và nó sẽ còn nghiêm trọng hơn cả những cái có thể có bên Toán học. Lập luận cho rằng một số học sinh yếu kém trong môn Toán học có thể sẽ khá hơn khi học Tin học không có nhiều cơ sở lí thuyết. Ngược lại, việc dẫn nhập các phương tiện Tin học có thể « cứu vãn » các học sinh có khó khăn và khuyến khích các em khác khi học toán. (Báo cáo của uỷ ban nghiên cứu Dacunha-Castelle, 1989)

Mặt khác uỷ ban nghiên cứu Kanhe cũng phê phán việc dẫn nhập các phương tiện Tin học theo hướng này là không có tham vọng dạy các đối tượng Tin học :

> Chương trình Toán học những năm 90 quy định rằng học sinh phải làm chủ được việc sử dụng máy tính bỏ túi khoa học (ở lớp 10 và lớp 11) và máy tính bỏ túi lập trình (ở lớp 12). Nếu việc yêu cầu này hiện diện trong lời nói đầu của báo cáo, nó vắng mặt trong phần liên quan đến nội dung và các hoạt động được đề nghị. Làm sao mà các nhà viết sách có thể tưởng tượng được việcdạy học chúng ? Lúc này có lẽ là thời điểm thích hợp để huy động các phương tiện để mong muốn này có thể trở thành hiện thực. (Báo cáo của uỷ ban nghiên cứu Kahne, 2001)

Trong cùng thời gian đó, ở Việt nam các nhà Toán học và cũng đồng thời là các nhà trí quyển như Nguyễn Văn Trang và Tạ Duy Phượng (2000), đã phát biểu như sau trong một hội thảo về dạy và học môn Toán tổ chức bởi bộ Giáo dục và Đào tạo :

> Việc quan trọng nhất trong việc dạy học không phải là dạy các phép tính với các phép toán số học hay là dạy việc sử dụng các chương trình đã được tích hợp trong máy tính bỏ túi (như thống kê, giải các phương trình v.v…) mà là dạy học *tư duy thuật toán*. Tư duy thuật toán đóng một vai trò quan trọng trong kỉ nguyên công nghệ thông tin, được thể hiện qua nhiều bài tập mà nội dung *gắn liền một cách sâu sắc với*

các nội dung Toán học. […] Nhờ máy tính bỏ túi, tư duy thuật toán sẽ là mối liên hệ giữa hai bộ môn liên quan chặt chẽ với nhau nhưng lại được dạy một cách tách rời trong nhà trường, đó là Toán học và Tin học. […] Nhiều thuật toán (tìm các số nguyên tố lớn, tính toán các công thức truy hồi mà trước đó ta không thực hiện được khi chưa có máy tính bỏ túi (giới hạn, tích phân). Khi chuẩn bị giáo án sư phạm, điều đó sẽ giúp một phương pháp mới được phát triển, đó là việc kết hợp lí thuyết với thực hành. (Trang 30)

Tuy nhiên như ta thấy trong phát biểu này tư duy thuật toán được nhắc đến một cách chung chung và nó chưa chỉ ra được các tri thức cụ thể cần giảng dạy.
Ngược lại, trong khuôn khổ của việc « dạy học môn Toán học » uỷ ban nghiên cứu về giảng dạy Kahne đã chỉ ra các khái niệm cơ bản của lý thuyết thuật toán và lập trình :

- Đối với lập trình : các cấu trúc điều khiển (vòng lặp và rẽ nhánh) và đệ quy ;
- Đối với lý thuyết thuật toán : các cấu trúc dữ liệu và độ phức tạp của thuật toán ;

Bây giờ ta sẽ xem xét một cách chi tiết hơn các đối tượng được xác định như các khái niệm cơ bản của lý thuyết thuật toán và lập trình để làm rõ hơn, một mặt là sự khác biệt và thống nhất giữa lý thuyết thuật toán và lập trình, mặt khác là việc lựa chọn các đối tượng cơ sở để đưa vào DHTPT.

I. Một số nét khái quát về các khái niệm cơ sở của lý thuyết thuật toán và lập trình

Chương trình là một văn bản, câu chuyện mà nó kể là thuật toán, việc lập trình là cách viết thuật toán. (Ganascia, 1998)

I.1. Thuật toán

Trong từ điển bách khoa toàn thư, Hebeinstrait đưa ra định nghĩa sau đây cho từ « thuật toán » :

Thuật toán là một dãy hữu hạn các quy tắc cần áp dụng trong một thứ tự nhất định cho một số hữu hạn các dữ liệu, để sau một số hữu hạn bước có thể đi đến một kết quả, và điều đó không phụ thuộc vào các dữ liệu. (Hebeinstrait, Từ điển bách khoa toàn thư, tập 12, trang 306)

Lý thuyết thuật toán là khoa học về các thuật toán :

Đối tượng của lý thuyết thuật toán là thiết kế, đánh giá và tối ưu các phương pháp tính toán trong Toán học và trong Tin học. (Flajoret, Từ điển bách khoa toàn thư, tập 1, trang 814)

Các đối tượng cơ bản của lý thuyết thuật toán bao gồm :

- Các thuật toán cơ bản : thuật toán Toán học, các thuật toán sắp xếp, các thuật toán tìm kiếm, các thuật toán hình học, các thuật toán đồ thị v.v… ;
- Các cấu trúc dữ liệu : mảng, cây, tập hợp, danh sách, bản ghi v.v… ;
- Phân tích độ phức tạp của thuật toán, lớp các bài toán NP ;
- Các kĩ thuật thiết kế thuật toán : lập trình động, thuật toán tham lam v.v… ;

Như vậy lý thuyết thuật toán cố gắng tìm cách nghiên cứu thuật toán một cách độc lập với việc cài đặt nó trên một máy tính cụ thể.

Không đi vào các chi tiết Toán học cụ thể, ta có thể nói rằng khi ta nghiên cứu tính hiệu quả của một thuật toán (độ phức tạp của thuật toán), ta thường để ý đến việc nghiên cứu sự thay đổi của số các câu lệnh cơ sở trong sự phụ thuộc vào số lượng các dữ liệu cần xử lí hơn là thời gian thực hiện thuật toán cũng như là số lượng ô nhớ mà nó chiếm trong bộ nhớ. Việc dựa trên các tính toán về độ phức tạp của thuật toán liên quan đến thời gian thực hiện nó trên một máy tính nào đó không cho phép đề cập đến cấu trúc nội tại của thuật toán cũng như đặc tính kĩ thuật của máy tính : tùy theo công việc, tốc độ của bộ xử lý, tốc độ truy cập vào các dữ liệu và thậm chí cả việc chạy thuật toán (các yếu tố ngẫu nhiên có thể can thiệp vào quá trình này), thời gian chạy một thuật toán không phải lúc nào cũng như nhau. (Từ điển Wikipedia 2005)

Trong công việc thiết kế thuật toán, thường người ta cần phải tổ chức, sắp xếp và chỉ định các dữ liệu của bài toán. Trong số các cấu trúc dữ liệu cơ sở được giới thiệu trong các giáo trình

về Tin học đại cương ở bậc đại học như : danh sách, mảng, ngăn xếp, bản ghi, khái niệm *biến* đều có mặt và đóng một vai trò quan trọng.

I.2. Lập trình

Lập trình trong Tin học là hoạt động cho phép viết ra các chương trình Tin học nhằm cài đặt nó trên máy tính.

> Chương trình Tin học là danh sách các câu lệnh mà máy tính phải tuân theo, theo một trình tự thực hiện nhất định. […] Người ta nạp chương trình Tin học trong bộ nhớ của máy tính và máy tính sẽ xử lí các câu lệnh của chương trình trong quá trình chạy chương trình tuỳ theo tốc độ của nó. (Arsac, Từ điển bách khoa toàn thư, tập 19, trang 31)

Các cấu trúc điều khiển cho phép sự chỉ dẫn việc thực hiện một thuật toán trên máy tính. Trong các giáo trình nhập môn Tin học, người ta phân biệt ba cấu trúc điểu khiển trong lập trình có cấu trúc như sau :

- cấu trúc tuần tự : các câu lệnh được thực hiện nối tiếp ;
- cấu trúc rẽ nhánh : biểu diễn bởi các câu lệnh điều kiện ;
- cấu trúc lặp : lặp lại một nhóm câu lệnh mà ta gọi là vòng lặp ;

II. Lý thuyết thuật toán và lập trình trong dạy học môn Toán ở trường phổ thông

Một cách ngắn gọn, trong dạy học toán phổ thông (DHTPT), ta có thể nhận thấy :
- sự tồn tại của các thuật toán, nhưng các yếu tố của lý thuyết thuật toán không được giảng dạy (ở cả Pháp cũng như Việt nam) ;
- sự tồn tại của các chương trình tính toán cho các máy tính bỏ túi lập trình hoặc cho các bảng tính nhưng các yếu tố lập trình không được dạy (ở Pháp) ;
- sự tồn tại của các máy tính nhưng kiến trúc của các máy tính này không phải là đối tượng của việc dạy học (ở Pháp) ;

Để giải thích các nhận xét này, ta không thể hoàn toàn gán trách nhiệm cho sự yếu kém của những nhà làm chương trình như theo cách làm của ủy ban nghiên cứu Kanhe mà cần suy xét đến các điều kiện sinh thái của hệ thống didactic liên quan đến các đối tượng này theo quan điểm của Chevallard (1986) :

> Khi người ta cải cách chương trình thì các « đối tượng mới » cũng được đưa vào trong dạy học. Tuy nhiên, các đối tượng này thường tỏ ra « quá lớn ». Vì giáo viên nhanh chóng nhận ra rằng việc đưa chúng vào trong dạy học là nặng nề, không thích đáng, thậm chí không thể làm được, nên giáo viên thường phải « hồ », phải phân cỡ, phải lọc và thậm trí loại bỏ các đối tượng này. Vấn đề didactic đặt ra như vậy, theo một khía cạnh nào đó, đã được nhiều người biết. Nhưng để phát biểu chúng một cách đúng đắn chúng, chúng ta cần phải thống nhất rằng vấn đề này không chỉ xuất hiện có tính chất riêng lẻ (đó là tôi chưa nói đến ví dụ khá nổi tiếng về « đường thẳng afine được dạy ở lớp 8) mà thường có tính chất hệ thống. Cần phải thống nhất để nhìn nhận ở đó không chỉ là tác động của một vài quyết định vội vàng của các nhà làm chương trình mà thực sự là các hậu quả tất yếu của các quy luật đặc thù của sự vận hành didactic.

Điều ghi nhận này dẫn chúng tôi đến các câu hỏi nghiên cứu đầu tiên như sau :

Các điều kiện nào có thể cho phép các đối tượng mới như cấu trúc dữ liệu và cấu trúc điều khiển có thể tồn tại được trong DHTPT ? Đó là các đối tượng cơ sở nào ? Với các loại bài toán nào ? Với các tổ chức Toán học và didactic nào ?

Chúng tôi sẽ phát triển các câu hỏi này song song với việc lựa chọn ngay từ đầu :

- một lĩnh vực của Toán học : tính toán số ;
- một công nghệ Tin học : máy tính bỏ túi.

III. Tính toán số, thuật toán và công cụ tính

III.1. Tính toán số và thuật toán

Tính toán số là một lĩnh vực Toán học lí tưởng trong việc hình thành các thuật toán, nhất là các thuật toán lặp mà ở đó có sự lặp lại một bất biến tính toán (ví dụ như phép chia hai số nguyên, xấp xỉ nghiệm của một phương trình số v.v). Sự lặp này dựa trên sự rời rạc hoá các tập số và dựa trên sự xác định các hành động có thể thực hiện được và lặp lại được. Sự phát biểu công việc này nhằm phục vụ cho việc tìm ra một kết quả cụ thể hay cho việc chứng minh chúng sẽ hình thành nên các thuật toán.
Trong các tổ chức « praxéologie » được đưa vào trong DHTPT (ví dụ như xấp xỉ thập phân, lập bảng giá trị cho một hàm số thực v.v), trong đó có sự hiện diện của các thuật toán, sự lặp lại có thể được cụ thể hoá bằng các đối tượng Toán học khác nhau như các công thức hàm số, các dãy số và khái niệm truy hồi.

III.2. Tính toán số và công cụ tính toán

Một phần khá quan trọng của tính toán lặp có thể được trao cho một hay nhiều công cụ tính toán. Các công cụ này cho phép người vận hành có thể thực hiện công việc tính toán nhờ vào thuật toán được tường minh hay được kết tinh trong một công cụ tính toán, ví dụ như bảng số, bàn tính, máy tính bỏ túi không lập trình được, v.v. Khả năng có thể trao việc tính toán lặp như vậy được khai thác khi các nhà làm chương trình tìm cách đưa một công cụ tính toán mới trong DHTPT để nhấn mạnh việc thu được các kết quả cụ thể của các tính toán số.
Việc đưa công cụ tính toán vào các thể chế dạy học môn toán như DHTPT dẫn đến sự phát sinh có tính thể chế. Sự phát sinh này sẽ tổ chức lại các tri thức và các kĩ thuật tính toán số. Ngược lại, ta cũng có thể dự đoán rằng việc đưa các yếu tố của lý thuyết thuật toán và lập trình có thể làm thay đổi việc chủ thể hoá công cụ tính toán số và sự đảm trách của thể chế đối với việc thu được kết qủa cụ thể trong tính toán.

III.3. Lựa chọn khái niệm vòng lặp và biến

Tầm quan trọng của các thuật toán lặp trong DHTPT dẫn chúng tôi đến việc tập trung việc nghiên cứu vào khái niệm vòng lặp. Khái niệm này được coi như là một ứng cử viên cho khái niệm cấu trúc điều khiển và đến lượt nó, cấu trúc điều khiển này mang tính tổ chức cho các cấu trúc điều khiển khác. Việc lựa chọn này dẫn đến việc xem xét khái niệm biến theo quan điểm lập trình (theo kiểu lập trình chỉ định), khái niệm này có quan hệ mật thiết với khái niệm vòng lặp. Các lí do của sự lựa chọn này sẽ được chỉ ra một cách cụ thể hơn trong quá trình nghiên cứu.
Chúng tôi hoàn chỉnh thêm việc xây dựng vấn đề nghiên cứu bằng những câu hỏi sau đây :

Làm thế nào để việc đưa các yếu tố của lý thuyết thuật toán và lập trình có thể dựa trên việc tích hợp máy tính bỏ túi vào DHTPPT ? Làm thế nào để điều đó có thể làm thay đổi bản chất của một công cụ tính ? Làm thế nào để việc dẫn nhập này có thể dựa trên các tổ chức praxéologie của tính toán số ? Làm thế nào để việc dẫn nhập này có thể làm thay đổi chúng ?

IV. Lựa chọn khung lí thuyết và phương pháp nghiên cứu

IV.1. Phân tích máy tính bỏ túi như một công cụ

Các khái niệm « công cụ » và các hoạt động Toán học được công cụ hoá đóng vai trò trung tâm trong nghiên cứu này. Khoảng 15 năm nay, các khái niệm này là đối tượng của nhiều nghiên cứu trong chuyên ngành didactic toán, nhất là trong chủ đề tích hợp các đối tượng kĩ thuật phức tạp trong dạy học. Các nghiên cứu này phát triển theo hai hướng chủ yếu sau :

- hướng tâm lí học ở đó người ta phân tích và khái niệm hoá các hoạt động được công cụ hoá trên cơ sở hoạt động nhận thức của cá thể ;
- hướng nhân chủng học ở đó người ta phân tích các hoạt động được công cụ hoá trên cơ sở thực tiễn và lí thuyết được xây dựng, được tăng giá trị và được chuẩn hoá bởi một thể chế.

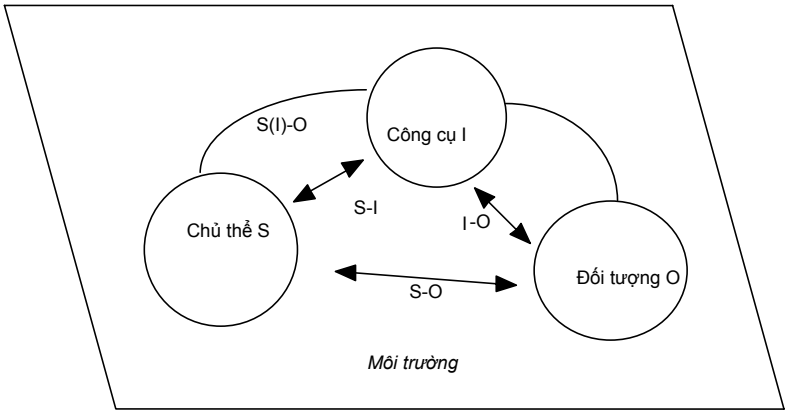
Mặc dù sử dụng hai khung lí thuyết tổng quát hoàn toàn khác nhau, hai hướng này mang đến cho dự án nghiên cứu của chúng tôi các cơ sở mà chúng tôi có thể khai thác và kết hợp chúng.

Theo quan điểm tâm lí học

Hướng nghiên cứu này trong lĩnh vực dạy học bắt nguồn từ các công trình về tối ưu hoá công cụ liên quan đến quá trình nhận thức (tham khảo Rabardel 1995 et Vérillon 1996).
Dưới đây là tóm tắt một bài báo của Artigue và Lagrange (1998) giới thiệu quan điểm này theo bốn ý cơ bản sau đây :

> 1. Một đối tượng kĩ thuật không trở thành một công cụ ngay lập tức thậm chí ngay cả khi người tìm cách coi nó như thế. Trước tiên nó chỉ là một đối tượng, hay là một dụng cụ nhân tạo, nếu ta dùng từ sử dụng bởi Rabardel (1996), nếu ta muốn giữ một vị trí trung lập nhất có thể có. Chỉ trong sự tiến hoá của mối quan hệ của chúng ta với đối tượng mà công cụ sẽ được hình thành trong một quá trình phát sinh công cụ phức tạp.
> 2. Sự phát sinh công cụ đồng thời hướng tới đối tượng và hướng tới chủ thể trong quá trình kép : *cá thể hoá công cụ* và *chủ thể hoá công cụ*. Cá thể hoá công cụ, hướng tới đối tượng, dẫn đến việc cá nhân hoá nó, biến đổi nó và có thể trao cho nó các chức năng mà một số trong đó không được tiên liệu bởi các nhà sản xuất công cụ này. […] Công cụ hoá, hướng tới chủ thể, dẫn đến việc soạn thảo hay tiếp thu một cách xã hội các dạng thức hành động được công cụ hoá.
> 3. Các dạng thức có nhiều chức năng. Đặt trong các tình huống cụ thể, chúng góp phần hiểu (chức năng *tri thức*), chúng góp phần phản hồi, biến đổi, giải quyết (chức năng *thực dụng*), và cuối cùng chúng góp phần tổ chức và điều khiển hành động (chức năng *orixtic*).
> 4. Các hoạt động được công cụ hoá ảnh hưởng đồng thời lên các kiểu tiếp cận tri thức và các tri thức được xây dựng bởi chính nó :
> - một mặt, nhờ các ràng buộc được dẫn nhập bởi dụng cụ, và ở mức độ này, dường như là cần thiết phải phân biệt các ràng buộc gắn liền với cấu trúc của dụng cụ (các ràng buộc nội tại), các ràng buộc gắn liền với các hành động hay sự biến đổi mà nó cho phép (các ràng buộc điều khiển), và cuối cùng là các ràng buộc gắn liền với cách thức mà dụng cụ hướng tới tổ chức các hành động (các ràng buộc tổ chức),
> - mặt khác, nhờ các tiềm năng mới tạo bởi các hành động của chủ thể với dụng cụ.
> Một cách tất yếu, điều đó dẫn đến quan hệ biện chứng giữa sự công cụ hoá và việc học môn Toán (Kỉ yếu của hội thảo pháp ngữ châu Âu, 1998, trang 18 và 19).

Chúng tôi đặc biệt chú ý đến khái niệm kép « cá thể hoá công cụ » và « chủ thể hoá công cụ ». Rabardel (1996) và Vérillon (1996) đã xây dựng khái niệm này bên trong sự mô hình hoá các **T**ình huống **H**oạt động được **C**ông cụ hoá. Mô hình này được các tác giả này gọi là THC (SAI theo tiếng pháp) và được mô tả trong hình vẽ dưới đây :



Vérillon (1996) giải thích như sau :

> Chủ thể hoá công cụ liên quan đến việc lập các quan hệ S-I : chủ thể phải xây dựng các dạng thức, các thủ tục, các phép toán cần thiết cho việc sử dụng dụng cụ. Ví dụ, chủ thể có thể hội nhập trong tình huống các

quan hệ S-I được xây dựng trong các hoàn cảnh khác và với các dụng cụ khác, hoặc ngược lại, xây dựng các quan hệ mới theo cách khám phá, hoặc theo cách bắt chước.
Cá thể hoá công cụ liên quan đến việc xây dựng các quan hệ I-O. Chủ thể gán cho công cụ khả năng hành động trên O và xây dựng các tính chất thuộc về chức năng mà chúng cho phép thực tại hoá khả năng hành động này. (Sách đã dẫn, trang 5)

Các thuật toán có tính chất lặp nằm ở trung tâm của sự phát triển các quan hệ mới đối với các đối tượng Toán học và Tin học mà chúng tôi sẽ tìm cách nghiên cứu.
Ví dụ đối tượng O mà Vérillon đề cập đến là đối tượng « biến Tin học » trong việc tính giá trị một biểu thức số, chủ thể S là một học sinh trong DHTPT và công cụ I là một máy tính bỏ túi hoặc là một bảng số. Quan hệ biện chứng chủ thể hoá công cụ / cá thể hoá công cụ mô tả phương thức mà ở đó sự hiện diện của công cụ sẽ ảnh hưởng vào sự hình thành của quan hệ S-O của học sinh. Sự hiện diện này có thể thay thế quan hệ S(I)-O, hay có thể làm phong phú quan hệ đó khi nó xuất hiện. Thậm chí sự hiện diện có thể tự áp đặt lên quan hệ này trong mọi tình huống mà máy tính bỏ túi được sử dụng. Chính theo cách hiểu này mà ta phải khẳng định rằng một công cụ không thể mang tính trung lập đối với việc tổ chức và thực hiện một tính toán số, cả về mặt tri thức luận cũng như mặt didactic. Công cụ không thể tích hợp một cách tự nhiên trong hoạt động tính toán của một cá thể. Theo Prud'home (1999) công cụ :

[…] là sự xây dựng hoạt động của nó trong các tình huống được giao. Công cụ không phải là thứ mà ta chỉ cần « đưa ra » sao cho cá thể kết hợp nó với hoạt động của mình. (Trang 29)

Quan điểm nhân chủng học

Theo Birebent (2001) :

Sự phát sinh công cụ mà Vérillon mô tả sẽ xuất hiện đối với mỗi cá thể như là một cái được xây dựng có đồng thời tính chất cá nhân và xã hội. Cái được xây dựng này phát triển tuỳ theo các tình huống có tính chất vấn đề với nó. Điều đó cho phép ta hình dung rằng các tính chất và mức độ đó rất khác nhau trong sự phát sinh công cụ cho cùng một lớp các bài toán. Nhưng trong một tình huống didactic của DHTPT, sự gặp gỡ của một học sinh với một bài toán không mang tính chất ngẫu nhiên. Theo một nghĩa nào đó, nó đã được chương trình hoá.

Chevallard (1992) đã bàn đến điều đó như sau :

Trong phần lớn các thực hành xã hội, đối tượng thường được « đánh giá » bởi kết qủa của hành động mà nó cho phép, và giá thành mà hành động này tạo ra. Trong thực tiễn của việc truyền thụ tri thức, hành động mà đối tượng cho phép nằm trong việc *thao tác một tri thức được dạy*, và nó tích hợp vào *quan hệ với tri thức* của các nhân vật chủ chốt của quan hệ didactic là người dạy và người học. Hoạt động này phải xuất hiện với tư cách hợp thức, và với tư cách này, nó đi vào khuôn khổ của các mục tiêu didactic quan trọng. Không có gì là bí ẩn cũng như là tai tiếng khi ta thấy một chiếc máy tính bỏ túi nằm trong tay của người bán hàng gia vị, của bà nội trợ hay của một học sinh nhưng ở bên ngoài lớp học. Và cũng như thấy nó, ở cùng một thời gian đó, tạo ra sự nghi vấn đối với nhiều giáo viên bộ môn (tất nhiên là không phải đối với các giáo viên bộ môn, chẳng hạn như Địa lí hay Vật lí vì đối với các bộ môn này máy tính bỏ túi có một tư cách tri thức luận và didactic đã được đảm bảo, đó là tư cách trợ giúp cho việc tính toán). Trong một trường hợp cụ thể, và thậm chí trong đa số trường hợp, việc sử dụng nó không đặt ra một vấn đề nào cả (trừ các vấn đề liên quan đến vật chất : giá, tính vận hành, tính thao tác v.v…). Trong trường hợp khác, việc sử dụng nó kéo theo nhiều vấn đề và các giáo án sư phạm gắn liền với việc tích hợp của công cụ này ở trong lớp học Toán là đối tượng của nhiều nghiên cứu, nhiều thực nghiệm và nhiều hội thảo. Người ta đã bao giờ thấy các chủ hàng bán gia vị tổ chức hội thảo liên quan đến đề tài này ?

Cũng vậy, Rabardel (1995) mô tả :

Chủ thể thay thế một dụng cụ nhân tạo bởi một công cụ khi trao cho dụng cụ này tư cách là một phương tiện để đạt được mục đích của hành động.

Cũng như Birebent, chúng tôi cho rằng sự thể chế hoá ở mỗi cá thể là kết quả của một sự thay thế mà ở đó cá thể là chủ thể và là người xác định hành động. Chính trong thể chế DHTPT mà học sinh khám phá ra các nhiệm vụ tính toán và cùng với các nhiệm vụ này là một hay nhiều kĩ thuật, cũng như một hay nhiều phát biểu lí thuyết.

IV.2. Phân tích có tính chất so sánh các thể chế

Trong nghiên cứu này chúng tôi sẽ tiến hành so sánh hai hệ thống dạy học ở phổ thông và hai hệ thống dạy học bậc đại học ở Pháp và ở Việt nam : DHTPT và dạy học môn Tin học ở bậc đại học.
Chúng tôi sẽ tiến hành so sánh theo hai chiều, lịch đại và đồng đại. Phân tích lịch đại cho phép hiểu rõ hơn tình trạng hiện tại của một hệ thống từ sự tiến hoá của hệ thống này.
So sánh đồng đại có mục đích :

- Phát hiện và đặc tính hoá các kết qủa khác nhau cũng như giống nhau của chuyển đổi didactic của cùng một đối tượng tri thức ;
- Đặt các câu hỏi cho sự giống nhau cũng như sự khác nhau liên quan đến các điều kiện và các ràng buộc của hệ thống ;
- Khởi tạo một thư mục của các tổ chức praxéologie tồn tại và dự tính sự phát triển có thể xảy ra của chúng.

> Đối diện với một sự bắt buộc phải hành động, chúng ta thường bắt đầu bởi việc quan sát và phân tích […] các phương thức mà người khác làm […]. Tiếp theo, chúng ta sẽ đánh giá những gì mà sự quan sát cũng như phân tích có thể phát hiện ra […], và sau đó tiến hành các giải pháp riêng của chúng ta bằng cách thử hoàn thiện thêm trên một số điểm mà chúng ta đánh giá là chưa tốt. (Chevallard 1998)

IV.3. Công nghệ didactic

Trong nghiên cứu này, theo một phương pháp nghiên cứu đặc thù của môn didactic toán, chúng tôi đã thiết kế và thực nghiệm một công nghệ didactic. Công nghệ didactic đồng thời là sự thực hiện didactic trong lớp học và cũng là phương tiện nghiên cứu được xây dựng một cách biện chứng với các phân tích tri thức luận và thể chế được tiến hành trước đó. Công nghệ didactic này dẫn đến sự phát sinh thực nghiệm từ việc kết hợp các biến didactic trên một tình huống cơ sở (Brousseau 1998) cho các tri thức Tin học : biến và vòng lặp.

> Một tập hợp các bài toán có thể được phát sinh bắt đầu từ một tình huống nhờ sự thay đổi các giá trị của một vài biến nào đó. Các biến này, đến lượt chúng lại làm thay đổi các đặc tính của các chiến lược cho lời giải (giá thành, tính hợp thức, độ phức tạp v.v) […]
> Chỉ có các sự thay đổi ảnh hưởng đến thứ bậc của các chiến lược là được xét đến (các biến thích đáng) và trong số các biến thích đáng, những biến mà giáo viên có thể điều khiển được đặc biệt quan trọng : đó là các biến didactic. (Brousseau 1982)

Công nghệ didactic này đã được thực nghiệm trong DHTPT ở cả hai nước. Việc xây dựng công nghệ này đã được manh nha từ luận văn Thạc sĩ của chúng tôi (xem Nguyen 2002) và trải qua một quá trình thai nghén lâu dài xen kẽ bởi các tiền thực nghiệm ở Pháp cũng như ở Việt nam.
Công nghệ didactic này cho phép ta có thể hành động một cách duy lí trên mỗi hệ thống dạy học (dựa trên các kiến thức didactic đã được tiền thiết lập) và cho phép việc kiểm chứng các giả thuyết nghiên cứu trong thực tế.
Quá trình thực nghiệm đã dẫn đến việc gặp gỡ các đối tượng phức tạp của thực tế dạy học và việc lí giải các hành xử của giáo viên. Do chưa tồn tại một khung lí thuyết đặc thù nên điều đó được tiến hành bởi các thoả thuận, phỏng vấn có tính chất tương đối không chính thức được thực hiện trước và sau thực nghiệm (các phỏng vấn sau thực nghiệm sẽ được ghi âm).

Việc lựa chọn chủ đề cho công nghệ didactic « Dẫn nhập một số yếu tố của thuật toán và lập trình trong DHTPT » làm chúng tôi gặp tình huống mâu thuẫn sau :

- Việc cổ xuý mạnh mẽ của các thể chế cho chủ đề này ;
- Sự lo lắng biểu lộ bởi các giáo viên.

Thật vậy, nếu như tất cả các giáo viên ở bậc THPT được liên hệ đã cho chúng tôi thời lượng giảng dạy của họ để thực hiện công nghệ didactic, một vài người trong số đã từ chối đứng lớp cho việc thực hiện thực nghiệm.

Do đó đối với chúng tôi công nghệ này mở ra một hướng mới liên quan đến vấn đề đào tạo giáo viên sao liên quan tới sự khuyến khích của các thể chế trong việc dẫn nhập các yếu tố Tin học trong DHTPT.

V. Cấu trúc của luận án

Nghiên cứu của chúng tôi được tổ chức theo cấu trúc sau

Phần A. Lý thuyết thuật toán và lập trình : Tổng quan về các nghiên cứu didactic đã thực hiện ; Phân tích thể chế
Chương A1. Các công trình chủ yếu trong didactic Toán lien quan đến thuật toán và lập trình;
Chương A2. Sự hiện diện của các khái niệm thuật toán, biến và vòng lặp trong DHTPT ở Pháp và Việt nam ;
Chương A3. Dự định đưa Tin học vào DHTPT ở Việt nam ;
Kết luận phần A

Phần B. Thuật toán và lập trình : một nghiên cứu tri thức luận
Chương B1. Sự xuất hiện của dạy học thuật toán và lập trình ;
Chương B2. Máy tính và chương trình ;
Chương B3. Từ chương trình Ada đến ngôn ngữ lập trình bậc cao ;
Kết luận phần B

Phần C. Thuật toán và lập trình : Lựa chọn biến didactic vĩ mô và biến didactic vi mô cho một công nghệ didactic
Chương C1. Một tình huống cơ sở cho thuật toán và lập trình : lập bảng của một hàm số nhờ máy tính ;
Chương C2. Máy tính nào cho DHTPT ? Hình thành cho việc thiết kế một máy tính có tính đại diện Alpro ;
Kết luận phần C

Phần D. Thiết kế, thực hiện và phân tích công nghệ didactic
Chương D1. Phân tích tình huống 1 : sự hiện diện và sự khác nhau của các biến nhớ A, B, C và phím nhớ Ans ;
Chương D2. Phân tích tình huống 2 : chương trình cho máy tính (Alpro, CALCULATOR) gần giống với máy tính giải tích của Babbage ;
Chương D3. Phân tích tình huống 3 : sự phát triển đồng thời của ngôn ngữ lập trình và máy tính Alpro, CALCULATOR II), máy tính Von Neumann ;
Chương D4. Nghiên cứu định tính hai nhóm học sinh xuyên suốt công nghệ didactic ;

Kết luận tổng quan và triển vọng nghiên cứu

VI. Các kết quả chính của nghiên cứu

VI.1. Kết quả của phân tích thể chế

Các phân tích này cho phép đưa ra các kết qủa dưới đây :
1. Cả hai thể chế đều có ý định rõ rệt là đưa các yếu tố Tin học vào trong DHTPT nhưng với các cấp độ khác nhau.
Ở Việt nam, việc tổ chức dạy học được đề nghị là trước tiên đưa khái niệm thuật toán như là đối tượng của tri thức với đầy đủ định nghĩa, tính chất và ví dụ. Sau đó nêu ra sự hoạt động của máy tính. Máy tính điện tử chỉ được nêu lên mà không hiện diện thực sự trong quá trình dạy học.

Ở Pháp, trong dạy học, người ta muốn gắn các khái niệm cơ sở của Tin học (như vòng lặp, biến, cấu trúc dữ liệu) với các phân môn của Toán học như giải tích, thống kê và mới đây là số học. Khái niệm thuật toán không phải là đối tượng dạy học.
Trong cả hai trường hợp, vị trí của học sinh đều khá hạn chế như là chúng tôi đã chỉ ra khi phân tích các bài tập dành cho học sinh.

2. Trong cả hai thể chế ta có thể nhận thấy sự thất bại của ý định đưa các yếu tố Tin học vào trường phổ thông nhưng với các cấp độ khác nhau.
Ở Việt nam, ta thấy là chương liên quan đến Tin học ở lớp 10 đã bị xoá bỏ và dự định đưa Tin học vào dạy học phổ thông như là một môn học độc lập cũng không tồn tại lâu. Ở Pháp, các khái niệm cơ bản của Tin học vẫn còn tồn tại ở trường phổ thông nhưng các khái niệm này đang ở trong một tình trạng khó khăn : điều đó được thể hiện bởi sự từ chối của các sách giáo khoa trong việc xây dựng các dạng nhiệm vụ liên quan đến lập trình các thuật toán. Hơn nữa sách giáo khoa luôn nhờ đến sự trợ giúp của các phần mềm để công cụ hoá các kĩ thuật liên quan đến các nhiệm vụ này.
Do đó phân tích của chúng tôi đã khẳng định các dự đoán bi quan mà các nghiên cứu trước đã tiên liệu trong việc dẫn nhập các yếu tố Tin học trong DHTPT.

3. Ta có thể đưa ra đây một số lí do cho thất bại này.
Ở Việt nam, chương liên quan đến Tin học bị biệt lập trong DHTPT (chúng tôi dùng lại từ của Lê Van 2001), vì các thuật toán lặp cần thiết cho phần này hiện diện rất ít trong chương trình Toán được dạy. Hơn nữa, sự vắng mặt hay việc không tính đến các công cụ tính toán (máy tính bỏ túi, máy tính điện tử) làm cho các thuật toán lặp này không bao giờ cho ra được kết quả cụ thể ngoài một vài tính toán bằng tay cho những vòng lặp đầu tiên (tính toán bằng tay).
Ở Pháp, trong DHTPT tồn tại nhiều nơi cư trú cho các thuật toán tính toán lặp như trong giải tích, thống kê và số học. Các nhà đề xướng chống lại cuộc cải cách toán hiện đại đã mong muốn sự có mặt của các thuật toán này nhằm đưa vào đó các hoạt động thực nghiệm trong dạy học Toán học. Đối với các nhà trí quyển này, một điều kiện để các thuật toán này cũng như các hoạt động thực nghiệm có thể tồn tại được là sự hiện diện của các công cụ tính toán. Hiện nay các phần mềm như bảng tính (trong máy tính bỏ túi hay máy tính điện tử) được khuyến cáo sử dụng và gánh vác trách nhiệm của các tính toán lặp và một phần của việc xây dựng các thuật toán.

Vượt lên trên cả các sự khác nhau trong vấn đề động cơ của thể chế, các điều kiện vật chất (thiết bị Tin học có mặt trong lớp học), nội dung và các tổ chức didactic, chúng tôi nghĩ rằng một phân tích mang tính chất tri thức luận về các tri thức nhắm tới, nhất là các khái niệm về biến và vòng lặp, sẽ cho phép phân tích sâu hơn các nguyên nhân của thất bại này.

VI.2. Kết qủa của phân tích tri thức luận

Phân tích tri thức luận đã cho phép xác định trong lịch sử Tin học và việc dạy học bộ môn một số điều kiện thuận lợi cho sự phát sinh các khái niệm biến và vòng lặp.
Các điều kiện được phát hiện ra như vậy dẫn đến việc tiến hành các lựa chọn cho việc thiết kế một công nghệ didactic nhằm dẫn nhập không chỉ các đối tượng cơ sở của thuật toán và lập trình mà còn là các khái niệm về máy tính và ngôn ngữ lập trình. Các lựa chọn và điều kiện này sẽ được kiểm nghiệm qua việc thực hiện công nghệ didactic và sau đó qua các phân tích kết quả thực hiện công nghệ.

Lựa chọn cho chiến lược dạy học các yếu tố của lý thuyết thuật toán và lập trình

Việc phân tích các sách chuyên luận cho phép làm nổi bật hai chiến lược dạy học sau :

- Sự hiện diện của một máy tính tham chiếu, có thể tồn tại thực hay giả tưởng. Các máy tính giả tưởng có thể là ảo (nghĩa là không được sản xuất, ví dụ như máy tính của Babbage, 1842), hoặc là lí tưởng (nghĩa là không được sản xuất nhưng đại diện cho

một lớp các máy tính tồn tại thực, ví dụ như máy tính MIX của Knuth, 1968). Do đó dạy học thuật toán sẽ gắn liền với việc hiểu được kiến trúc của máy tính cũng như với việc thiết kế một ngôn ngữ thích hợp cho máy tính này.

- Sự vắng mặt của một máy tính tham chiếu : người ta nhắm đến một ngôn ngữ lập trình đại diện cho một lớp các ngôn ngữ tồn tại (ví dụ như ngôn ngữ SPARKS của Horowitz, 1978). Điều đó giả định rằng người học phải có sẵn kiến thức tiên quyết về thuật toán và lập trình.

Sự vắng mặt trong thể chế dạy học phổ thông (ở Pháp cũng như ở Viêt nam) của tất cả các ngôn ngữ lập trình và sự phân tích các chiến lược dạy học dẫn chúng tôi đến lựa chọn cơ bản sau đây.

Giả thuyết 1

Do sự vắng mặt của tất cả các ngôn ngữ lập trình, việc dạy học các yếu tố thuật toán và lập trình phải tính đến vấn đề didactic chủ đạo là kiến trúc máy tính và mối quan hệ của nó với một ngôn ngữ lập trình (giống như trong chiến lược dạy học thứ nhất).

Lựa chọn các thông báo đến máy tính có bộ nhớ xóa được

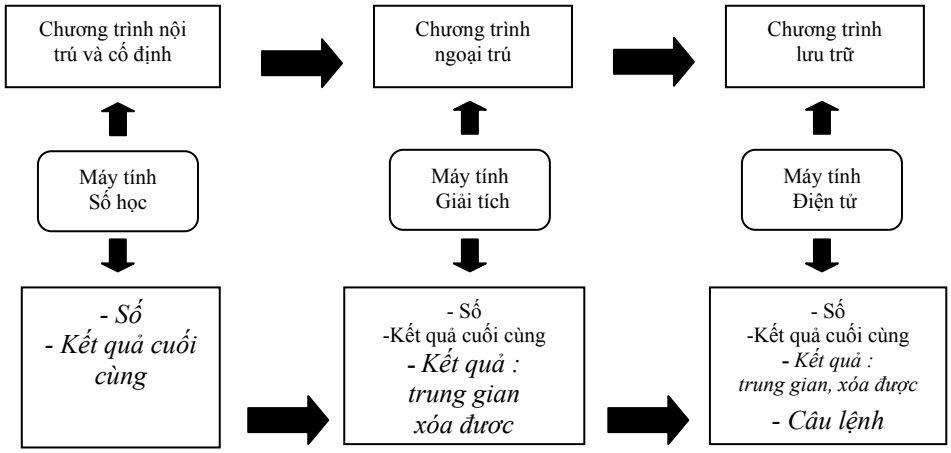
Trên quan điểm kiến trúc máy tính, ba giai đoạn "máy tính Số học", "máy tính Giải tích" và "máy tính điện tử" một mặt được đánh dấu bởi sự tăng tiến của quá trình lưu giữ trong bộ nhớ và mặt khác bởi tính chất cơ bản là sự xoá được của bộ nhớ. Tính chất này kết hợp với khả năng lưu giữ của máy tính làm tăng khả năng tính toán của máy tính.

Sự tiến triển của bộ nhớ dẫn đến khả năng lưu giữ và có thể xoá các kết quả trung gian (máy tính của Babbage) là một trong các nhân tố làm cho việc thực hiện một cách cơ học các tính toán lặp, tức là các thuật toán lặp, trở thành hiện thực.

Ý tưởng về chương trình lưu trữ và các dữ liệu có thể được lưu trên cùng một bộ nhớ (máy tính điện tử Von Neumann) cho phép xử lí một chương trình giống như các dữ liệu cho tính toán. Một chương trình, một khi được thực hiện, có thể trở thành một câu lệnh cho một chương trình khác.

Việc thực hiện một chương trình của một thuật toán lặp, tức là có vòng lặp, cần có sự phối hợp giữa tính toán (bộ tính trong máy tính) và các kết quả trung gian (bộ nhớ). Sự tồn tại của bộ điều khiển, hiện diện trong máy tính của Babbage và của Von Neumann, là nhân tố thứ hai cho phép ta có thể giao phó cho máy tính khả năng ra các quyết định : bắt đầu, dừng lại hay liên kết một tính toán với các tính toán khác. Điều này dẫn đến kết quả cơ bản là cho phép ta có thể viết một chương trình cho máy tính.

Chúng tôi sơ đồ hoá sự tiến triển này trong hình 1 dưới đây :



Hình 1. Tiến triển của bộ nhớ và của lập trình theo các kiểu máy tính

Viết một thông báo cho máy tính điện tử Von Neumann

Máy tính điện tử Von Neumann với khái niệm chương trình lưu trữ cho phép dịch từ tất cả các ngôn ngữ gần với ngôn ngữ tự nhiên (về mặt ngữ nghĩa) sang ngôn ngữ mã hoá và có thể thực hiện bởi máy. Nhắc lại rằng một ngôn ngữ như vậy là ngôn ngữ bậc cao.
Điều đó dẫn đến là có một sự xa rời dần dần các ngôn ngữ máy để tiến gần tới ý nghĩa của một thuật toán cần phải lập trình nhờ ba cấu trúc cơ bản : tuần tự, rẽ nhánh và lặp.

Giả thuyết 2

Việc viết một thông báo cho một máy tính Von Neumann để đạt được việc thực hiện một thuật toán lặp là một điều kiện thuận lợi cho việc hình thành trong hành động một số yếu tố của ngôn ngữ lập trình bậc cao : tuần tự, rẽ nhánh và lặp.

Ý nghĩa của khái niệm biến Tin học

Trong việc viết một chương trình Tin học, một ngôn ngữ được mã hoá (ví dụ như ngôn ngữ máy) quá gần với các đặc tính kĩ thuật của máy tính sẽ che dấu ý nghĩa của thuật toán được lập trình. Trong một ngôn ngữ lập trình, việc viết một vòng lặp của thuật toán lặp dẫn đến phải gán liên tiếp nhiều giá trị cho bộ nhớ có thể xoá được. Khái niệm phép gán cho phép giải thích một cách đầy đủ ý nghĩa của khái niệm biến Tin học : một biến Tin học là một ô nhớ xoá được và được chỉ định để nhận các giá trị liên tiếp.

Giả thuyết 3

Việc viết một chương trình cho tính toán lặp (thuật toán lặp) trong ngôn ngữ gần với sự hoạt động của máy tính là một điều kiện thuận lợi cho việc hình thành khái niệm biến như là một ô nhớ xóa được.
Đặc biệt, sự phân biệt có tính chất kí hiệu học giữa đẳng thức và phép gán có thể làm tăng cường sự phân biệt giữa đẳng thức Toán học và phép gán.

Sự cần thiết của hoạt động Toán học

Ví dụ có mang tính lịch sử của chương trình Tin học đầu tiên viết bởi Ada chứng tỏ rằng hoạt động Toán học là cần thiết cho việc lập trình từ một lời giải Toán học đã biết.
Hoạt động Toán học có tính chất thuật toán này phải cho phép đồng thời xác định một bất biến của vòng lặp và việc hình thành điều kiện dừng.

Giả thuyết 4

Việc hình thành điều kiện dừng và bất biến của sự lặp lại, vốn cần thiết cho việc viết một thông báo đến một máy tính có bộ nhớ xóa được, sẽ bắt buộc các hoạt động Toán học có tính chất suy ngẫm trên các đối tượng Toán học có mặt trong lời giải Toán học của một bài toán.

VI.3. Kết quả của phân tích các lựa chọn biến didactic vĩ mô cho công nghệ didactic

Việc dẫn nhập các đối tượng của thuật toán và lập trình trong hệ thống dạy học hiện nay sẽ thông qua các sự lựa chọn vĩ mô didactic và vi mô didactic trong mối quan hệ với các giả thuyết nghiên cứu. Các giả thuyết này sẽ được kiểm nghiệm nhờ sự thực hiện didactic.
Chúng tôi quan tâm đến việc thiết lập các biến vĩ mô didactic, nghĩa là các biến điều khiển mà chúng tôi có thể ra quyết định hành động trên nó và chúng « liên quan đến tổ chức tổng thể của công nghệ ». (Artigue 1988, trang 291)
Lựa chọn thứ nhất liên quan đến một tình huống cơ sở của lý thuyết thuật toán và lập trình : bằng việc điều khiển các giá trị của biến didactic của tình huống này, ta có thể tạo ra một sự phát sinh thực nghiệm đồng thời cho cả các khái niệm biến Tin học và vòng lặp nhờ các thông báo liên tiếp cho các máy có các đặc tính khác nhau.
Tình huống cơ sở được lựa chọn kết hợp với một bài toán Toán học là lập bảng giá trị cho một hàm số thực. Chúng tôi đã chỉ ra vai trò Toán học của vấn đề này trong việc hình thành và phát triển của các loại máy tính và ngôn ngữ lập trình. Bài toán Toán học này được kết hợp với một bài toán Tin học là viết một thông báo cho một máy tính để đạt được một sự thực hiện thông báo đó trên máy tính và cho ra kết quả thực.
Một cách cụ thể hơn, chúng tôi đã giới hạn bài toán Toán học vào việc lập bảng giá trị của một hàm số đa thức.

Một tình huống cơ sở của lý thuyết thuật toán và lập trình

Cho **f** là một hàm số đa thức bậc **n**. Tính các giá trị của hàm số tương ứng với **m** số x_0, x_1…x_{k-1}…cách đều nhau một khoảng **p** và thuộc vào đoạn **[a ; b]** ở đó $\mathbf{x_0 = a}$.

Bài toán Toán học này là một bài toán bình thường trong trường hợp bậc của hàm số nhỏ hơn hoặc bằng 2 và khi m lấy giá trị nhỏ. Một biến didactic mấu chốt của bài toán này là độ lớn của khoảng cách p so với độ rộng của đoạn [a ; b] vì độ lớn này xác định số m : trong trường hợp mà p rất nhỏ so với hiệu (b – a), việc tính toán bằng tay trở nên rất « tốn kém », thậm chí là không thể trong một khoảng thời gian bị giới hạn. Như vậy sẽ đặt ra vấn đề phải thực hiện thuật toán lặp này bằng máy tính.

Vấn đề là ta sẽ thực hiện tính toán bằng loại máy tính nào ?
Ngay từ đầu chúng tôi đã quyết định chọn lựa một loại máy tính tối thiểu, đó là máy tính bỏ túi không lập trình được. Loại máy tính này hiện diện trong cả hai thể chế của Pháp và Việt nam và không gắn liền với một loại ngôn ngữ lập trình nào cả.
Phân tích của chúng tôi đã chỉ ra rằng một máy tính như vậy không vận hành trên các nguyên tắc của máy tính giải tích của Babbage cũng như của máy tính điện tử Von Neumann và nó cũng không phải là máy tính số học. Thật vậy, ở thời điểm này, tất cả các loại máy tính hiện diện trong cả hai thể chế đều có các bộ nhớ xoá được (điều đó là ngược lại với máy tính số học như máy tính Pascal). Trong số các bộ nhớ xoá được, chúng tôi đã phân biệt các phím nhớ A, B, C với phím nhớ Ans. Loại phím nhớ A, B, C có thể là các ứng cử viên cho khái niệm biến Tin học.
Nhưng sự hiện diện của các phím nhớ này chưa thể đảm bảo cho sự cá thể hoá công cụ cũng như sự chủ thể hoá công cụ các phím trong thực tế sử dụng máy tính của học sinh. Thật vậy trong thể chế, các kĩ thuật mang tính chất thể chế cho các tính toán được công cụ hoá bởi máy tính bỏ túi và đặc biệt là việc lập bảng giá trị của một hàm số, không đòi hỏi việc sử dụng các phím nhớ A, B, C.

Điều ghi nhận này dẫn chúng tôi đến đến việc thiết lập giả thuyết nghiên cứu sau đây :

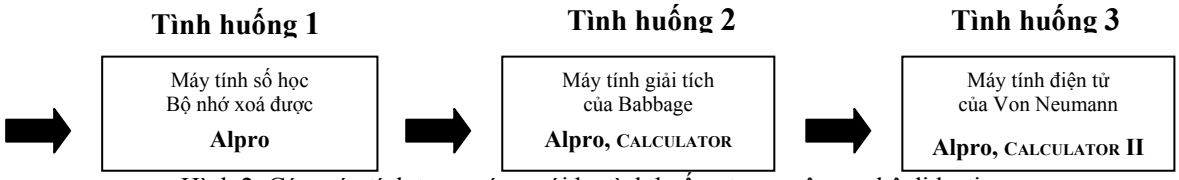
Giả thuyết 5

Trong DHTPT, máy tính bỏ túi vận hành như một máy tính số học, có nghĩa là một máy tính không có phím nhớ xoá được.

Như vậy điểm mấu chốt của công nghệ didactic là làm cho học sinh bước vào một sự phát sinh thực nghiệm. Sự phát sinh này sẽ dẫn dắt học sinh đi từ một máy tính số học (đối với họ là máy tính bỏ túi không lập trình được) đến máy tính điện tử Von Neumann.

Một phát sinh thực nghiệm

Tình huống cơ sở cho phép thiết kế công nghệ didactic như là một phát sinh thực nghiệm của máy tính Von Neumann và sự lập trình thông qua việc viết các thông báo liên tiếp (các chương trình) cho các máy tính có trang bị các đặc tính khác nhau. Chúng tôi mô tả dưới đây các loại máy tính khác nhau trong công nghệ didactic :



Hình 2. Các máy tính tương ứng với ba tình huống trong công nghệ didactic

Bài toán được lựa chọn, lập bảng giá trị của một hàm số, sẽ tìm cách phát động việc tính toán lặp thông qua thuật toán lặp hay đệ quy trong một thông báo cho máy tính có cơ chế hoạt động gần với máy tính Babbage. Nhưng số lượng hạn chế của các ô nhớ xoá được của máy tính Alpro sẽ ưu tiên cho thuật toán lặp thay vì thuật toán đệ quy.

Vấn đề mấu chốt của tình huống thứ nhất là việc làm quen với Alpro và các loại bộ nhớ khác nhau.

Vấn đề mấu chốt của tình huống thứ hai là việc thiết lập bất biến tính toán của thuật toán tính toán lặp trong thông báo cho máy tính có cơ chế hoạt động gần với máy tính của Babbage. Việc thiết lập này là điều kiện cần thiết cho việc khái niệm hóa của khái niệm biến Tin học.

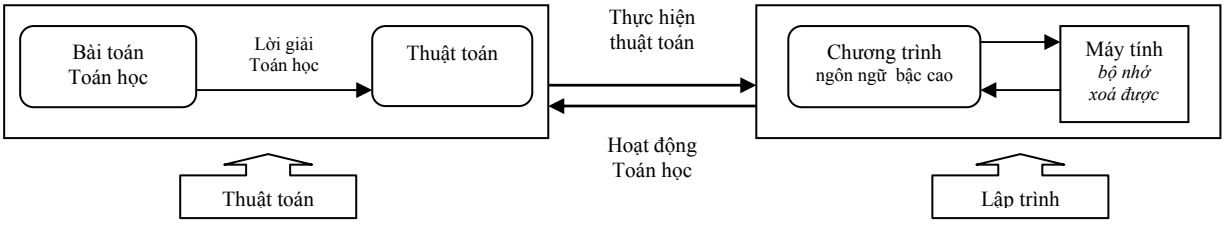
Vấn đề mấu chốt của tình huống thứ ba là viết thông báo cho máy tính điện tử Von Neumann. Việc viết này sẽ liên hệ với việc thiết lập điều kiện dừng, phép gán các biến Tin học cũng như các yếu tố của một ngôn ngữ lập trình cho máy tính điện tử.

Trong ba tình huống này, học sinh có hai vị trí khác nhau :

- Một vị trí liên quan đến khía cạnh thuật toán : học sinh làm việc trên các thuật toán, đó là lời giải Toán học của bài toán lập bảng số ;
- Vị trí kia liên quan đến khía cạnh lập trình : học sinh viết các chương trình cho các máy tính lần lượt khác nhau.

Trong cả hai vị trí trên, học sinh sẽ nhận các thông tin và các tương tác sinh ra từ việc chạy chương trình của mình trên các máy tính thực sự hoặc một phần nào đó giả tưởng.

Chúng tôi thể hoá chúng trên sơ đồ sau :



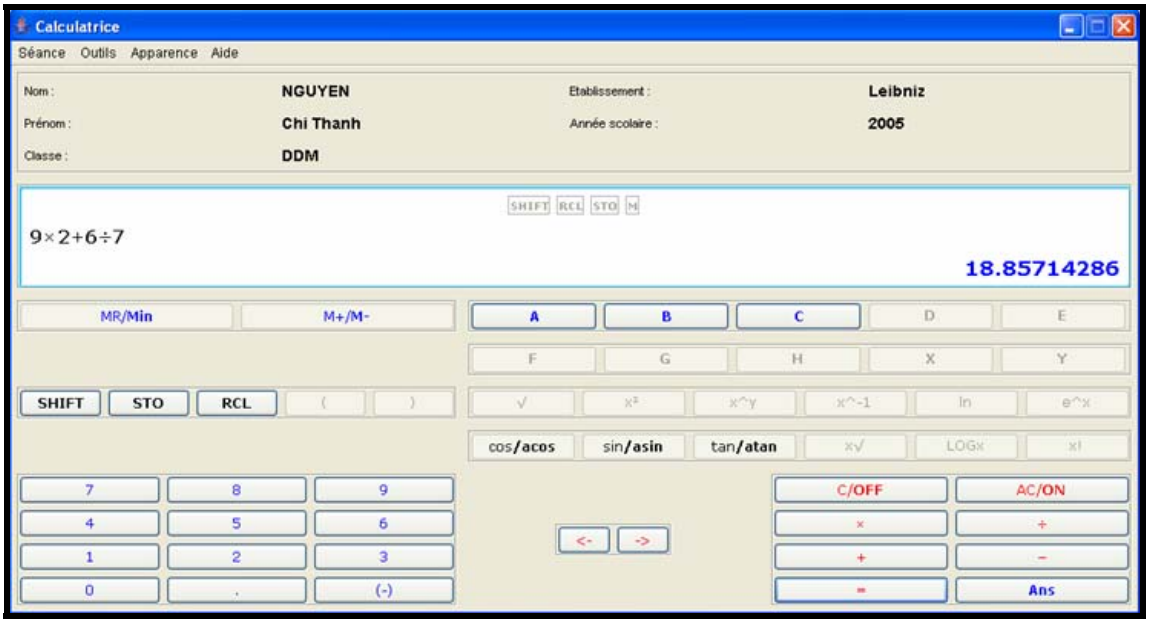
Hình 3. Hai vị trí « thuật toán » và « lập trình » của học sinh

Các lựa chọn cho các số và máy tính Alpro

1. Các số mà ta phải tính giá trị của chúng qua hàm số f thuộc vào tập **Dj**

Dạng viết thập phân của các số trùng với dạng hiển thị của kết quả của một phép toán trên màn hình của máy tính bỏ túi bình thường. Sự lựa chọn này cho phép dẫn nhập việc tính toán trong khuôn khổ của phép tính với kết qủa thực giống như các tính toán hiện diện trong hai thể chế.

2. Máy tính bỏ túi không lập trình được Alpro, máy tính cơ sở của công nghệ didactic
Do nhu cầu của nghiên cứu, chúng tôi đã thiết kế một mô phỏng của máy tính bỏ túi, tên là Alpro. Máy tính mô phỏng này dựa trên mô hình của các máy tính có mặt trong hai thể chế và có thêm khả năng là ghi lại lịch sử của các dãy bấm phím cho một tính toán nào đó.
Trong công nghệ, chúng tôi đã khoá một số phím. Dưới đây là giao diện của phiên bản này, gọi là Alpro thu hẹp.



Hình 4. Hình ảnh tổng quan của máy tính mô phỏng Alpro, phiên bản thu hẹp

Các phím bị khoá là tất cả các phím màu xám.

3. Các lựa chọn cũng như các lí do của chúng được trình bày chi tiết dưới đây.

Bộ nhớ Ans và việc hạn chế các bộ nhớ biến chỉ còn ba

Phân tích của chúng tôi đã làm rõ sự tồn tại của hai loại phím nhớ nhìn thấy được : phím biến nhớ A, B, C và phím nhớ kết quả cuối cùng Ans. Để cho tiện, chúng tôi sẽ gọi là bộ nhớ biến và bộ nhớ Ans.
Hai loại bộ nhớ này có cấp độ khác nhau khi xét đến mức độ nhìn thấy được và mức độ xoá được. Hai bộ nhớ này dành để nhận các số cho một tính toán có kết quả thực và có thể dùng lại được trong tính toán. Trong một bộ nhớ biến, việc lưu trữ một số phải mang tính chất chủ định. Trong khi đó một số được ghi một cách tự động trong một bộ nhớ Ans ngay khi mà phím « = » (hay phím « Sto ») được ấn. Một số được lưu trữ một cách có chủ định trong một bộ nhớ biến sẽ bị xoá khi mà người sử dụng quyết định lưu trữ một số mới vào đó ; trong khi đó một số lưu trữ trong bộ nhớ Ans sẽ bị xoá ngay khi mà phím « = » (hoặc « Sto ») được bấm.
Cũng giống như máy tính bỏ túi không lập trình được, máy tính Alpro có hai loại bộ nhớ, bộ nhớ biến và bộ nhớ Ans. Nhưng cũng khác với các loại máy tính này, Alpro chỉ có ba bộ nhớ A, B, C để thúc đẩy việc sử dụng các bộ nhớ biến A, B, C cũng như việc hình thành khái niệm biến Tin học. Các phím liên quan đến các bộ nhớ biến như « Sto » và « Rcl » cũng có mặt ; người ta có thể bấm chúng trực tiếp mà không cần thông qua phím « Shift ».

Khoá các phím ngoặc

Trong phiên bản được sử dụng cho công nghệ didactic, chúng tôi đã khoá các phím ngoặc (bộ nhớ không nhìn thấy) để làm thuận lợi hoặc làm cho việc sử dụng các bộ nhớ (A, B, C hay Ans) trở nên cần thiết khi thực hiện các tính toán cũng như khi lập trình việc tính toán giá trị của của một số qua một hầm số.

Sự hiện diện chỉ cho các phím của bốn phép toán số học

Liên quan đến các phím tính toán, chúng tôi đã quyết định chỉ giữ lại bốn phép toán số học « + », « - », « × », « ÷ ». Quyết định này sẽ làm tăng số lần bấm phím, đặc biệt đối với phép nhân vì nếu không sử dụng bộ nhớ, để tính chẳng hạn v^n, cần phải vào số v n lần. Việc tăng số lần bấm phím sẽ làm tăng nguy cơ nhầm lẫn trong tính toán và do đó chứng thực cho việc sử dụng các bộ nhớ.

Có thể truy cập các thông tin liên quan đến các phím nhớ

Máy tính mô phỏng Alpro cho phép truy cập thông tin liên quan đến các phím nhớ khi dịch chuyển con trỏ trên giao diện của Alpro gần với các phím này.

Ví dụ khi dịch chuyển con gần với phím « A », thông tin « A : lưu trữ một giá trị trong bộ nhớ A ». Cũng vậy, việc chuyển con trỏ trên phím « Ans » cho phép thông tin sau « Ans : Gọi lại kết quả cuối cùng được tính ».

Các thông tin này có thể khởi động quá trình công cụ hoá một phím nhớ biến hoặc phím nhớ Ans.

VII. Kết luận tổng quan của luận án và hướng phát triển

VII.1. Các kết quả chính của luận án

Các khái niệm cơ sở trong Tin học như vòng lặp và biến được hình thành đồng thời với sự tiến triển của kiến trúc máy tính. Đây chính là ý tưởng xuyên suốt công nghệ didactic và điều đó thu được nhờ các phân tích thể chế và khoa học luận. Ý tưởng này có thể được thể hiện như sau : « Từ Alpro - máy tính số học - đến Alpro - bộ tính toán trong máy tính điện tử Von Neumann - ».

Đặc tính xoá được của bộ nhớ và khái niệm biến

Khái niệm biến Tin học chỉ được phản ánh đầy đủ khi mà một bộ nhớ, vượt ra ngoài khả năng lưu trữ thông thường (ở đây là các con số), được quan niệm như là bộ nhớ xoá được. Điều khẳng định này được chứng tỏ một cách đồng thời bởi các phân tích xuất phát từ sự hình thành lịch sử của máy tính điện tử cũng như sự phát sinh thực nghiệm trong cả hai thể chế (Việt nam và Pháp) qua các tình huống của công nghệ didactic. Nhưng khái niệm này, để có thể tồn tại được trong dạy học toán được cần được cụ thể hoá như là bộ nhớ của một máy tính, có nghĩa là một vị trí trong máy tính dành cho việc lưu trữ dữ liệu.

Bản chất chủ đạo của các bộ nhớ ở cuối tình huống thứ nhất của công nghệ mang tính chất của các biến Toán học : một bộ nhớ chứa một và chỉ một số trong quá trình tính toán. Do vậy một bộ nhớ chưa mang tính chất xoá được và do đó chưa phải là một biến Tin học. Việc thực hiện một chương tình tính trên Alpro cũng như lần gặp gỡ đầu tiên với các bộ nhớ trong pha khám phá các tính toán quen thuộc dường như đóng một vai trò mấu chốt cho quá trình chủ thể hoá công cụ các phím Alpro.

Việc lưu trữ các số vào các bộ nhớ có tính chất *chủ định* chỉ xảy ra vào cuối tình huống thứ nhất của công nghệ didactic. Đây là kết qủa rõ ràng nhất liên quan đến quá trình cá thể hoá công cụ các phím nhớ ở cuối tình huống thứ nhất. Với lí do này phím Ans, phím nhớ được sử dụng nhiều nhất, trở thành phím nhớ nhìn thấy được.

Cần phải đợi đến lúc kết thúc của công nghệ didactic để phần lớn các chương trình cuối cùng viết cho việc thực hiện một thuật toán lặp trên máy tính (Alpro, CALCULATOR II) mới có thể chứng kiến sự hiện diện của thuộc tính xoá được có tính chủ định qua phép toán cập nhật một biến. Các phím nhớ này do đó chiếm lĩnh được đầy đủ vai trò của biến Tin học.

Sự vận hành của máy tính như là máy tính số học

Các máy tính bỏ túi mà học sinh sử dụng trong DHTPT đều được trang bị các phím nhớ gắn liền với các bộ nhớ xoá được. Tuy nhiên phân tích thể chế và phân tích *a posteriori* của tình huống thứ nhất trong công nghệ didactic đã làm rõ một chức năng được đa số học sinh sử dụng đó là coi máy tính bỏ túi như là máy tính số học tức là các phím nhớ được sử dụng mà không được gắn liền với thuộc tính xoá được.

Đó cũng chính là kết quả của phân tích tri thức luận ở đó chúng tôi đã phân biệt máy tính số học với máy tính có bộ nhớ xoá được. Sự phát minh của máy tính có bộ nhớ xoá được đã cho phép Ada viết chương trình Tin học đầu tiên (Ada 1842).

Các máy tính bỏ túi mà học sinh trong DHTPT sử dụng có hai loại bộ nhớ : Ans, bộ nhớ ghi kết quả cuối cùng và bộ nhớ biến. Chỉ có bộ nhớ biến là có đủ tư cách để hỗ trợ việc quan niệm hoá khái niệm biến.

Việc sử dụng bộ nhớ Ans đã được nói đến từ các phân tích thể chế và được chi tiết hơn qua phân tích *a posteriori* của tình huống thứ nhất. Chúng tôi đã rút ra được hai tính chất đặc thù của thể chế ở Pháp liên quan đến các tính toán được công cụ hóa[1].

- Một quy tắc của hợp đồng sư phạm : học sinh được phép lấy các kết quả của việc tính toán là các giá trị gần đúng bằng cách làm tròn hoặc cắt cụt các số hiển thị trên màn hình máy tính bỏ túi ;
- Một cách sử dụng máy tính bỏ túi mang tính chất cá nhân ở đó một tính toán được chia thành các tính toán nhỏ hơn. Việc chia này được cho phép bởi việc lưu trữ *không mang tính chủ định* các kết quả cuối cùng trong bộ nhớ Ans (chúng tôi đặt tên là Ans~~Sto~~).

Chúng tôi thậm chí có thể khẳng định rằng sự phát sinh có tính công cụ của máy tính bỏ túi mà nó đóng vai trò như máy tính số học, ở Pháp cũng như ở Việt nam chưa hoàn toàn kết thúc. Thật vậy việc chủ thể hoá công cụ các phím nhớ biến A, B, C và phím nhớ Ans phụ thuộc vào việc chủ thể hoá các tổ hợp phím phức tạp, ví dụ như AC/ON, C/OFF, « = », « Sto » mà ta có thể thấy sự khó khăn trong việc làm chủ các chức năng của chúng qua việc phân tích thực nghiệm.

Sự tối ưu việc giao tiếp người – máy và khái niệm biến

Sự hiện diện của các tính toán lặp kéo theo nó sự tồn tại của các thuật toán lặp là một trong các nguyên nhân lí giải cho sự quan tâm của chúng tôi đối với khái niệm vòng lặp. Nguyên nhân còn lại là sự xuất hiện của chương trình Tin học đầu tiên viết bởi Ada (1842) cho máy tính có bộ nhớ xoá được (máy tính giải tích của Babbage) để thực hiện một thuật toán lặp.

Chúng tôi đã lập giả thuyết rằng việc chủ định thực hiện các thuật toán lặp với một bất biến tính toán là điều kiện cần để thiết kế và thiết lập sự tối ưu trong việc giao tiếp với máy tính. Khái niệm vòng lặp là một câu trả lời cho việc tìm kiếm này.

Chính giả thuyết này làm cơ sở cho quá trình chuyển đổi từ tình huống thứ hai sang tình huống thứ ba, từ người máy CALCULATOR sang người máy CALCULATOR II, người máy này lưu trữ các chương trình được truyền đạt. Vì lí do này, máy tính (Alpro, CALCULATOR II) là

[1] Chính trong tình huống thứ nhất mà sự phân tích so sánh giữa Pháp và Việt nam đã làm rõ các thực tiễn khác nhau giữa hai nước (cho cấp bậc sau trung học cơ sở). Đối với tình huống thứ hai và thứ ba, ta có thể nói rằng tình trạng hiện tại của hai hệ thống đều có sự giống nhau về việc tảng lờ các đối tượng Tin học cơ sở như vòng lặp và biến.

một máy tính hoạt động theo nguyên lí chương trình lưu trữ, do đó là máy tính điện tử Von Neumann.
Đó cũng chính là kết quả của phân tích tri thức luận ở đó chúng tôi đã phân biệt máy tính với chương trình lưu trữ (máy tính điện tử Von Neumann) với máy tính có bộ nhớ xoá được (máy tính giải tích Babbage).

Sự tiến triển của ngôn ngữ Alpro đến ngôn ngữ cho máy tính điện tử Von Neumann

Máy tính điện tử Von Neumann có tích hợp một chương trình dịch và được lưu trữ trong bộ nhớ. Điều đó sẽ cho phép việc dịch tất cả các ngôn ngữ gần với ngôn ngóithong thường về mặt ngữ nghĩa ra một ngôn ngữ có thể được thực hiện trực tiếp bởi máy tính. Điều đó cho phép sự tiến triển của các ngôn ngữ từ ngôn ngữ máy đến ngôn ngữ cấp cao.
Đồng thời với việc xuất hiện khái niệm vòng lặp, chúng tôi đã thiết kế bước chuyển từ tình huống thứ hai đến tình huống thứ ba, (từ việc lập trình một thuật toán lặp cho máy tính có bộ nhớ xoá được đến việc lập trình một thuật toán lặp cho máy tính điện tử Von Neumann). Bước chuyển này yêu cầu học sinh tham gia vào sự tiến triển của ngôn ngữ lập trình : học sinh phải bổ xung thêm vào các câu lệnh tuần tự, có mặt ngay từ tình huống thứ nhất, các câu lệnh lặp cũng như các câu lệnh để thoát khỏi vòng lặp.
Điều đó được đặc biệt nhấn mạnh cho việc cập nhật các biến thông qua phép gán trong ngôn ngữ Alpro. Việc cập nhật các biến vốn gắn bó chặt chẽ với quá trình chủ thể hoá các phím nhớ xoá được A, B, C phải liên hệ với việc khởi tạo biến để phục vụ cho việc viết các vòng lặp.

Hoạt động Toán học cần thiết cho việc lập trình một thuật toán

Phân tích tri thức luận đã chỉ ra rằng Ada, người đầu tiên viết chương trình cho máy tính, đã phải biến đổi một thuật toán Toán học « vô hạn » sang một thuật toán lặp hữu hạn nhằm viết một chương trình cho máy tính giải tích của Babbage.
Các phân tích định tính một số học sinh cho thấy rằng học sinh đã tiến hành các hoạt động trên các thuật toán liên quan đến việc lập bảng giá trị hàm số để thiết lập, để xây dựng bất biến tính toán cũng như điều kiện dừng của vòng lặp. Việc viết này bắt buộc sự làm việc có tính chất suy ngẫm trên các đối tượng Toán học hiện diện trong bài toán lập bảng giá trị hàm số như sự biến thiên của hàm số, sự rời rạc hoá của tập xác định của hàm số.

Một dạng nhiệm vụ mới

Công nghệ didactic đã đưa ra một dạng nhiệm vụ mới trong Tin học « viết một chương trình Tin học để sản sinh ra các kết quả cụ thể bắt đầu từ một thuật toán lặp (được biết đến như một lời giải Toán học) ». Phân tích thể chế đã chỉ ra sự vắng mặt của dạng nhiệm vụ này trong DHTPT.
Phân tích *a posteriori* đã chứng tỏ rằng ta có thể uỷ thác dạng nhiệm vụ này cho học sinh. Phân tích cũng chỉ ra rằng đối với việc viết một thuật toán lặp, ta có thể thiết lập một sự cộng tác giữa giáo viên và học sinh trong quá trình thể chế hoá các đối tượng Tin học cơ sở (như vòng lặp, biến, các phép toán trên các biến), công việc này được duy trì với việc hợp thức hoá kết quả mang tính chất cú pháp trên máy tính Alpro.

Tính chất hai mặt của máy tính trong công nghệ didactic

Sự vắng mặt của tất cả các ngôn ngữ lập trình trong chương trình phổ thông (ở Pháp cũng như ở Việt nam) và khả năng có thể chọn lựa một trong hai chiến lược dạy học mà chúng tôi đã làm rõ trong phân tích các sách chuyên luận cho phép chúng tôi quyết định việc dựa vào máy tính trong công nghệ didactic (chiến lược dạy học được nêu trong sách chuyên luận của Knuth 1968).

Các máy tính hiện diện trong công nghệ didactic có hai mặt, mang tính thực tại cho máy tính Alpro và mang tính giả tưởng cho người máy CALCULATOR.
Sự tiến hoá của người máy đóng một vai trò mấu chốt : từ bộ điều khiển không thể lặp lại một phép toán, trong máy tính (Alpro, CALCULATOR) trở thành bộ điều khiển có thể lưu trữ chương trình (chương trình dịch), trong máy tính điện tử Von Neumann (Alpro, CALCULATOR II).
Tính chất hai mặt này của các máy tính được chọn lựa cho phép sự vận hành việc hợp thức có tính chất cú pháp : hợp thức thực dụng bởi việc chạy chương trình tính toán trên Alpro (tình huống thứ nhất), hợp thức mang tính chất khái niệm theo các tiêu chuẩn cho các chương trình có vòng lặp cho máy tính điện tử của Von Nemann. Các tiêu chuẩn này là kết quả của sự cộng tác (học sinh với học sinh hoặc học sinh với giáo viên) cho việc viết chương trình trong ngôn ngữ cấp cao (tình huống thứ hai và thứ ba).

VII.2. Hạn chế của nghiên cứu và những hướng đi mới của nghiên cứu

1. Công nghệ didactic dừng lại khi các đối tượng Tin học cơ sở như biến và vòng lặp mới được định hình. Như vậy chưa có sự ổn định các kiến thức này trong việc thực hiên các kiểu nhiệm vụ cần được sáng tạo ra.
Chỉ có một tập giá trị được gán cho biến (kiểu của biến), đó là các số viết dưới dạng thập phân. Thế mà trong DHTPT ta thấy sự tồn tại của các kiểu cấu trúc dữ liệu khác như mảng và danh sách.
Một câu hỏi mới được đặt ra :

Làm thế nào để tiếp tục công nghệ didactic nhằm đưa lại nghĩa « Tin học » cho các cấu trúc cơ sở này.

Công nghệ didactic ưu tiên cho việc thiết kế vòng lặp mà ở đó số lần lặp được tính toán trước. Điều đó cũng có nghĩa là công nghệ didactic cố định kiểu của điều kiện dừng : vòng lặp « lặp lại n lần…». Tuy nhiên phân tích đã chỉ ra là các kiểu vòng lặp khác cũng tồn tại trong DHTPT như : « chừng nào…làm » hoặc « lặp…đến khi… ». Sự lựa chọn của chúng tôi trong việc ưu tiên vòng lặp đầu tiên bị ràng buộc bởi tính chất của Alpro.
Việc thu hẹp các kiểu vòng lặp vào chỉ một loại vòng lặp « lặp lại n lần… » sẽ hạn chế ý nghĩa của khái niệm này trong Tin học.
Việc lựa chọn thuật toán lặp cũng hạn chế ý nghĩa của một khái niệm Tin học khác gắn bó chặt chẽ với khái niệm vòng lặp : đó là điều kiện rẽ nhánh.

Liệu có cần phải cải tiến người máy Alpro để nó có thể lưu trữ được các chương trình có chứa điều kiện dừng ?
Hay chỉ cần thay đổi Alpro bằng cách thêm các phím « logic » ví dụ như quan hệ thứ tự « > » (hay nhiều quan hệ thứ tự) để mở ra thêm các khả năng cho việc viết vòng lặp ?

Đó là một số hướng mà chúng tôi muốn mở ra để tiếp tục hay chỉnh đổi công nghệ didactic.

2. Mô phỏng Alpro của máy tính bỏ túi có một số hạn chế khác : ví dụ, phím « = » vốn đã có sẵn nhiều ý nghĩa trong các tính toán đại số hay số. Việc sử dụng máy tính bỏ túi đã bổ xung thêm một ý nghĩa nữa đó là ghi một cách tự động một kết quả vào trong bộ nhớ Ans.
Trong công nghệ didactic, chúng tôi đã áp đặt thêm một ý nghĩa nữa, đó là câu lệnh in một kết quả được chọn lựa.
Chúng tôi nghĩa rằng việc lựa chọn này đã có thể là nguồn gốc của một số khó khăn liên quan đến việc cập nhật các bộ nhớ.
Ta có thể sửa chữa khá dễ dàng điều bất lợi này bằng cách thêm vào Alpro một chức năng in điều khiển bởi một phím « in ».

3. Chúng tôi đã chọn bài toán lập bảng giá trị của một hàm số, bài toán này tồn tại trong DHTPT để cho học sinh làm lập trình, loại vấn đề này không tồn tại trong DHTPT. Tuy nhiên vấn đề này được đảm trách một cách thích đáng bởi bảng tính trong DHTPT ở Pháp.
Thật vậy trong tiền thực nghiệm, một vài học sinh lớp 11 ban khoa học (1er S) đã đề nghị lời giải cho vấn đề này bằng cách xử dụng bảng tính có trong các máy tính của học sinh.

Liệu có thể đề cập đến việc tiến hành lại công nghệ didactic với Alpro (có thể được cải tiến) với các vấn đề trong Toán học trong lĩnh vực số học hay thống kê mà ở đó không có các phần mềm có thể đảm trách việc lặp ?

4. Trong các phân tích định tính một số học sinh, chúng tôi đã nghiên cứu sự phát sinh có tính chất công cụ các phím Alpro, chúng tôi đã không xét đến sự phát sinh này liên quan đến Alpro với tư cách là bộ tính toán trong hai loại máy tính, một là máy tính có bộ nhớ xoá được (Alpro, CALCULATOR) và sau đó là máy tính điện tử Von Neumann (Alpro, CALCULATOR II).

Trên máy tính và trong một tình huống didactic, với điều kiện nào ta có thể thiết lập một nghiên cứu như vậy ?

5. Một kết quả mà Ravel (2003) đã thu được trong việc ghi nhận sự khó khăn làm sống được một số yếu tố của thuật toán và lập trình trong DHTPT khi nghiên cứu về dạy học số học ở lớp 12 ban khoa học ở Pháp (Ter S) :

> Các khó khăn trong việc làm sống được tính chất thuật toán của số học qua việc sử dụng máy tính bỏ túi, bảng tính hay là các phương tiện Tin học khác được thể hiện qua việc lựa chọn của giáo viên. Cũng như các sách giáo khoa, giáo viên rất ít sử dụng các dụng cụ Tin học để thực hiện một hoạt động thực sự cho các bước tiến hành có tính chất thuật toán trong số học. (Ravel 2003, trang 266)

Khó khăn này được bộc lộ một phần qua cách mà giáo viên tham gia thực nghiệm tìm cách kiểm tra kết quả thu được khi sử dụng máy tính. Việc kiểm tra này được làm dựa vào thói quen của tính toán nhẩm và cách ước lượng kết qủa trung gian và kết qủa cuối cùng.
Chúng tôi minh họa điều này bằng trích đoạn dưới đây từ pha tổng kết trong tính toán thứ ba.

> 21. Giáo viên (Gv) : Như vậy liệu các em có thể làm một vài phép thử ? **Tôi để các em tính toán bằng tay, các em lấy nháp ra, ta thử làm nhanh điều đó mà không cần đặt các phép tính. Bằng cách nhẩm, hãy thử ước lượng một giá trị gần đúng thô của y và sau đó là của z**. Làm theo hai bước. Bây giờ ta thử xem, **một phẩy gì đó, ta có thể thay thế bằng bao nhiêu ?**
> 22. Nhiều học sinh cùng một lúc : Bằng 1.
> 23. Gv : 1, đúng rồi, tại sao lại không ? Em nói là bao nhiêu ?
> 24. Hs : //
> 25. Gv : **Gần bằng 4 ?** Em có thể nêu phân số mà em thu được ?
> 26. Hs //
> 27. Gv : **Gần bằng 16 chia 4 ?**
> 28. Hs : Đúng rồi.

Cùng lúc đó, giáo viên viết lên bảng :

$$y \approx \frac{16}{4} \approx 4$$

> 29. Gv : **Gần bằng 4. Liệu sai số có lớn không khi ta thay 1,257 bằng 1 ?**
> 30. Hs : Có.
> 31. Hs : Không
> 32. Gv : **Liệu có sự thay đổi về dấu không ? Liệu có nguy cơ thay đổi dấu hay cái gì đó tương tự không ?** Nếu như đó là 1,3 chẳng hạn. Chín lần 1,3 sẽ là bao nhiêu ? Chín lần 1,3, ví dụ vậy ? Nó gần bằng 11. Cuối cùng là 9 lần 13, sẽ là kém 11 hoặc kém 12. Như vậy phần ở trên tử đó sẽ luôn là số dương. Nó không làm thay đổi dấu, đúng không ? **Và ở dưới mẫu, liệu có thay đổi gì cơ bản không ?**

Như vậy chính thực tế giảng dạy của giáo viên trong việc tính nhẩm đã thiết lập tiêu chuẩn để hợp thức hoá kết qủa trong việc dạy tính toán với máy tính số học : ta nhận ra ở đó có sự đảo lộn tổ chức praxéologie dẫn đến việc xây dựng lại việc nhìn nhận vấn đề liên quan đến « ý tưởng bao trùm của các tính toán không được công cụ hoá ».
Sự kiểm tra mang tính chất thuật toán các tính toán sẽ kéo theo nó các kĩ thuật và các lí thuyết mới (mà ta cần sáng tạo ra trong DHTPT ?) khác với kĩ thuật và lí thuyết được đề nghị bởi giáo viên. Những cái này không khác gì việc lấy lại các kĩ thuật mang tính chất số học và đại số có trước sự hiện diện của các máy tính mà ta đã thấy trong ví dụ trên.

Làm thế nào để xây dựng các tổ chức praxéologie mới cho kiểu nhiệm vụ mới liên quan đến việc tính toán dụng cụ hoá này ? Liệu ta có thể dựa trên các tổ chức đang tồn tại trong DHTPT ?

Công nghệ didactic của chúng tôi do đó đã gợi mở ra nhiêu câu hỏi mới. Theo chúng tôi, các câu hỏi này đóng vai trò trung tâm trong việc đào tạo giáo viên môn Toán liên quan đến các đối tượng cơ sở của Tin học trong mối liên quan với các vấn đề Toán học. Việc đào tạo này cũng là sự hưởng ứng với các đề nghị được nhấn mạnh của các nhà trí quyển trong việc cổ xuý việc đưa các đối tượng này vào dạy học phổ thông.
Phát triển công nghệ didactic (bao gồm cả mô phỏng máy tính Alpro được cải tiến) để tích hợp cho việc đào tạo giáo viên là một hướng nghiên cứu ưu tiên mà chúng tôi mong muốn tiếp tục trong tương lai.

Bibliographie

ADA L. (1842), *Notes of "Sketch of the Analytical Engine invented by Charles Babbage"*, Bibliothèque de Genève, sur http://www.fourmilab.ch/babbage/sketch.html

AHO A., HOPCROFT J., ULLMAN J. (1989), *Structures des données et algorithmes*, (traduit en français par Moreau J-M), Edition InterEdition

ARSAC O. et **BOURGEOIS C.** (1987), *Option informatique 1re*, Edition Nathan

ARTIGUE M. (1988), « Ingénierie didactique » in *Recherches en didactique des Mathématiques* Vol. 9/3, Edition la Pensée Sauvage, (pp. 281-307)

ARTIGUE M. et **LAGRANGE J-B.** (1998), « Instrumentation et écologie didactique de calculatrices complexes : éléments d'analyse à partir d'une expérimentation en classe de Premières S » in *Actes du colloque francophone européen, Calculatrice symbolique géométrique dans l'enseignement des mathématiques*, Irem de Montpellier (pp. 15-38)

BARNEY et **CABRERA** (2003), *John von Neumann and von Neumann Architecture for Computers (1945)* sur http://www.salem.mass.edu/~tevans/VonNeuma.htm

BARON G-L. (1987), *La constitution de l'informatique comme discipline scolaire, le cas des lycées*, Thèse en sociologie de l'Education, Université René Descartes, Paris

BESSOT A., COMITI C. (2005), "Some comparative studies between France and Vietnam Curriculums" in *Mathematics Education in different traditions: A comparative study in Asian and Western countries*, Edition Kluwer

BESSOT A. (2003), *Modèles didactiques des situations d'apprentissage*, Cours polycopié en Master II en Didactique des Mathématiques, Université de Joseph Fourier, Grenoble

BESSOT A., NGUYEN C.T. (2005), « L'ingénierie didactique, l'exemple d'une recherche coopérative franco-vietnamienne » in *Actes du premier séminaire franco-vietnamien en Didactique des Mathématiques*, Juin 2005, Université de Pédagogie de Ho Chi Minh-ville

BERTRANDIAS J-B. (1992), « Mathématiques et Informatique » in *L'ordinateur pour enseigner les mathématiques* (sous la direction de Cornu B.), Edition Puf (pp. 71-95)

BIREBENT A. (2001), *Articulation entre la calculatrice et l'approximation décimale dans les calculs numériques de l'enseignement secondaire français – Choix des calculs*

trigonométriques pour une ingénierie didactique en classe de Première scientifique, Thèse en Didactiques des Mathématiques, Université Joseph Fourier, Grenoble 1

BIREBENT A., NGUYEN C.T. (2003), « Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement des mathématiques à l'aide de la calculatrice », texte existant sous forme électronique dans les *Actes de la XIIè Ecole d'été de didactique des mathématiques*, Edition La Pensée Sauvage - Grenoble

BIREBENT A., NGUYEN C.T. (2005), « Conception d'une ingénierie didactique pour l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement secondaire des mathématiques » in *Les cahiers Leibniz*, N° 125 (pp. 1-8) (disponible sur http://www-leibniz.imag.fr)

BOLLIET L. (1993), « Jean Kuntzmann (1912-1992) : un extraordinaire pionnier » in *Troisième colloque sur l'histoire de l'Informatique en France*, Sophia-Antipolis, (5 p.)

BÖHM et **JACOPINI** (1966), "Flow diagrams, Turing machines and language wit only two formation rules" in *Communications of the ACM, Vol. 9, N°5,* (pp. 366-371)

BRETON P. (1990), *Une histoire de l'informatique*, Edition Seuil

BREZINSKI C. (1988), *Algorithme numérique*, Edition Ellipses

BROUSSEAU G. (1982). « Les objets de la didactique des mathématiques » in *Actes de la Troisième école d'été de didactique des mathématiques*, Olivet

BROUSSEAU G. (1998), *Théorie des situations didactiques*, édité par Balacheff et al., Edition la Pensée Sauvage

CARDTION A. et **CHARRAS C.** (1996), *Introduction à l'algorithmique et à la programmation*, Edition Ellipses

CHABERT J-L et *al*. (1994), *Histoire d'algorithmes – Du caillou à la puce*, Edition Belin

CHEVALLARD Y. (1986), « Les programmes et la transpositions didactiques. Illusions, contraintes et possibles » in *Bulletin APMEP*, n° 352, (pp-. 32-50)

CHEVALLARD Y. (1991), « Dimension instrumentale, dimension sémiotique de l'activité mathématique » in *Séminaire de Didactique des mathématiques et de l'informatique*, LSD2 IMAG, Grenoble

CHEVALLARD Y. (1992), « Concept fondamentaux de la didactique : perspectives apportées par une approche anthropologique » in *Recherche en didactique des mathématiques*, N° 12/1, Edition La Pensée Sauvage, Grenoble (pp. 73 – 111)

CHION J. (1972), *Initiation à la programmation en langage Algol W*, Université Scientifique et Médicale de Grenoble

CORNU B. et **ROBERT. C.** (1983), *Mathématique et calculatrice programmable au lycée et au Baccalauréat*, Edition Magnard, Paris

CRDP CLERMONT-FERRAND (1981), *La machine arithmétique ou Pascaline*, Collection Documents Régionaux

DESCOTTES D. (2003), « Pascal et les nombres » in *Pour la Science*, N° 16, (pp. 18-31)

DIJKSTRA E-W. (1968), "Go To Statement Considered Harmful" in *Communications of the ACM, Vol. 11, N° 3*, (pp. 147-150)

DIJKSTRA E-W. (1970), *Concern for Correctness as a Guiding Principle for Program Composition*, Collected papers, sur http://www.cs.utexas.edu/users/EWD/

DISCALA R. M. (2004), *Essentiel de l'informatique et de la programmation*, Edition Berti

DUCATEAU C.F et **FLECK J.** (1993), « Quelques réflexions sur l'enseignement de la programmation en IUT » in *Troisième colloque sur l'histoire de l'Informatique en France*, Sophia-Antipolis, 1993 (12 pages)

DUPUIS C., EGRET M. A., GUIN D. (1988), « Présentation et analyse d'activités de programmation en Logo » in *Petit x*, N° 18 (pp. 47-69)

DURAND-RICHARD. M. J. (2003), *Des lois de pensée à l'intelligence artificielle* sur http://ufr6.univ-paris8.fr/lit-math/maths/DOXhist/LPIA0506.doc

FRADIN M., LETANG C. (1973), *Introduction à l'algorithmique*, Edition Technique et vulgarisation, Paris

GANASCIA J-G. (1998), *Dictionnaire de l'informatique et des sciences de l'information*, Edition Flammarion, Paris

GODFREY M.D., HENDRY D.F. (1993), "The Computers as Von Neumann planned it" in *IEEE Annals of the History of Computing*, Vol. 15, N° 1 (pp. 11-21)

GOLSCHLAGER et **LISTER** (1986), *Informatique et algorithmique*, InterEdition

GRANJON Y. (1999), *Informatique – Algorithmes en Pascal et en langage C*, Edition Dunod

GUILLIER F. (2005), *Histoire de l'informatique* sur http://histoire-informatique.org

HOANG N.T. et **PHAM T.N.** (1999), *Tin học đại cương* (Informatique générale), Edition « Nhà xuất bản Khoa học và kĩ thuật », Hanoï, Viêt-nam

HOPCROFT J. (1986), « Les machines de Turing » in *Les mathématiques aujourd'hui*, *Bibliothèque Pour la Science*, (pp. 157-168), Edition Belin

HOROWITZ E. et **SAHNI S.** (1978), *Fondamentals of Computer Algorithms*, Edition Pitman

HO S.Đ et al. (1990), *Tin học đại cương* (Informatique générale), Edition « Trường Đại học tổng hợp Hà nội » Viêt-nam

HUNAULT G. (2004), *Histoire des langages de programmation* sur http://info.univ-angers.fr/pub/gh/hilapr

IBM (1961), *Fortran II, General information manual*, International business machines corporation

IBM (1969), *Flowcharting techniques*, International business machines corporation

IFRAH G. (1994), *Histoire universelle des chiffres*, Edition Robert Laffont, Paris

KNUTH D-E. (1963), "Computer-Drawn Flowcharts" in *Communications of ACM, Volume 6, N°9*, (pp. 555-563)

KNUTH D-E. (1968), *Algorithmes fondamentaux, l'art de la programmation d'ordinateur*, Addison-Wesley Publishing Company

KNUTH D-E. (1970), "Von Neumann's first computer program" in *Computing Surveys Vol. 2, N° 4*, (pp. 247-259)

KNUTH D-E. (1971), *Seminumerical Algorithms,* Addison-Wesley Publishing Company

KNUTH D-E. (2005), *MMIX 2009 a RISC computer for the third millennium* sur http://www-cs-faculty.stanford.edu/~knuth/mmix.html

KUNTZMANN J. (1957), *Formation de calculateurs – Moyens de Calcul – L'atelier Arithmétique*, 2e édition, Université de Grenoble, Année 1957 – 1958

KUNTZMANN J. (1969) *Méthode numérique*, Collection Enseignement des sciences, Edition Hermann

KUNTZMANN J. (1988), « Naissance de l'informatique à l'université de Grenoble » in *Premier colloque sur l'histoire de l'Informatique en France*, IMAG Grenoble 1988

LABORDE C., BALACHEFF N., MEJIAS B. (1985), « Genèse du concept d'itération : une approche expérimentale » in *Enfance* N° 2-3, (pp. 223-239)

LABORDE J. (1967), *Cours pratique de langage Algol*, Edition Dunod Université

LAGRANGE J-B. (1992), « Représentations mentales des données informatiques et difficultés d'acquisition chez des débutants en programmation » in *Bulletin de l'EPI*, N° 67 (pp. 91-103), N° 68 (pp. 119-137)

LAGRANGE J-B. (2002), « Les outils informatiques entre « sciences mathématique » et enseignement, une difficile transposition ? » in *Calculatrice symboliques – transformer un outil en un instrument du travail mathématique : un problème didactique*, (pp. 89-116), Edition La Pensée Sauvage

LEHATEM S. (2004), *Rapport de stage de Master I*, Equipe DDM, Université de Grenoble

LE THAI BAO T. T. (2004), *Etude sur la notion de limite de fonction dans l'enseignement des mathématiques : ingénierie didactique dans un environnement calculatrice*, Mémoire de Master II en Didactique de Mathématiques de l'université de Pédagogie de Ho Chi Minh ville, Viêt-nam

LE THI H.C. (1997), *Etude didactique et épistémologique sur l'enseignement du vecteur dans deux institutions : la classe de dixième au Viêt-nam et la classe de sconde en France*, Thèse en Didactiques des Mathématiques, Université Joseph Fourier, Grenoble 1

LE VAN T. (2001), *Etude didactique de liens entre fonctions et équations dans l'enseignement des mathématiques au lycée en France et au Viet-nam*, Thèse en Didactiques des Mathématiques, Université Joseph Fourier, Grenoble 1

LIGONNERE R. (1987), *Préhistoire et histoire des ordinateurs*, Edition Robert Laffont, Paris

MARGUIN J. (1997), « Une histoire du calcul artificiel et de ses concepts » in *Sciences, Association française pour l'Avancement des Science*s, N° 97-1, 1997

MARSUARD C. (2003), *Petite synthèse à l'intention de ceux qui n'ont pas connu le Gamma 60*, sur http://febcm.club.fr/systemes/histoire_systemes.html

MEJIAS B. (1985), *Difficultés conceptuelles dans l'écriture d'algorithmes itératifs chez des élèves de collège*, Thèse en Mathématiques appliquées – option Didactique, Université Scientifique et Médicale de Grenoble

MENABREA L.F. (1842), *Sketch of The Analytical Engine Invented by Charles Babbage*, Bibliothèque de Genève sur http://www.fourmilab.ch/babbage/sketch.html#menaref2

MAYSONNAVE J. (1996), *Introduction à l'algorithmique générale et numérique*, Edition Masson

MOREAU R. (1987), *Ainsi naquit l'informatique,* Edition Dunod, Paris

NEYRET R. (1995), *Contraintes et déterminations des processus de formation des enseignants*, Thèse en Didactique de Mathématiques, Université Joseph Fourier, Grenoble

NGUYEN BA K, VU D.T. (1992), *Phương pháp dạy học môn Toán* (Méthodologie de l'enseignement des mathématiques), Edition « Nhà xuất bản Giáo dục », Hanoi, Viêt-nam

NGUYEN BA K, LE KHAC T. (1993), *Dạy học một số yếu tố của Toán học tính toán và Tin học* (Enseignement quelques éléments de l'Analyse numérique et de l'Informatique), Edition « Nhà xuất bản Giáo dục », Hanoi Viêt-nam

NGUYEN C. T. (2002), *La notion d'algorithme dans l'enseignement des mathématiques au lycée. Comment l'émergence des notions de boucles et de variables s'articule à des connaissances en mathématiques* ? Mémoire de Master II en EIAH-D, Université Joseph Fourier, Grenoble

NGUYEN C. T., BESSOT A. (2003), « La prise en compte des notions de boucle et de variable informatique dans l'enseignement des mathématiques au lycée » in *Petit x* N° 62, (pp. 7-32)

NGUYEN THE T., VUONG DUONG M. (2001) « Dạy học sử dụng máy tính bỏ túi » (Enseignement en utilisant la calculatrice) in *Actes du colloque Méthodologie de l'enseignement de Mathématique secondaire* (pp. 74-84), Ministère de l'Education et de la Formation du Vietnam

NGUYEN THI NHU H. (2004), *La calculatrice de poche dans l'enseignement – apprentissage des mathématiques au Viêt-nam – Le cas du système d'équations du premier degré à deux inconnues en classe de 10^{e}*, Mémoire de Master II en Didactique de Mathématiques, l'université de Pédagogie de Ho Chi Minh ville, Viêt-nam

NGUYEN XUAN H. (1988), *Thuật toán* (Algorithmique), Edition « Nhà xuất bản thống kê », Hanoi, Viêt-nam

NGUYEN VAN T., TA DUY P. (2001), « Kì thi giải toán trên máy tính bỏ túi, 5 năm nhìn lại » (Le concours de la résolution des problèmes mathématiques sur la calculatrice, un regard rétrospectif sur cinq dernières années) in Actes du colloque *Quelques réflexions pour une amélioration de la pratique de calcul sur la calculatrice de la marque Casio* (pp 25-31), Ministère de l'Education et de la Formation du Viêt-nam

NGUYEN VAN T. et *al.* (2003), *Hướng dẫn thực hành toán trên máy tính bỏ túi Casio Fx-500MS, Fx-570MS* (Guide pratique sur la résolution des problèmes mathématiques sur la calculatrice Casio Fx-500MS, Fx-570MS), Département de l'enseignement secondaire du Ministère de l'Education et de la Formation au Viêt-nam

PRUDHOMME G. (1999), *Le processus de conception de système mécaniques et son enseignement. La transposition didactique comme outil d'analyse épistémologique,* Thèse en Didactiques des Mathématiques, Université Joseph Fourier, Grenoble 1

QUACH TUAN N. (2002), *Ngôn ngữ Pascal* (Langage de programmation Pascal), Edition « Nhà xuất bản thống kê », Hanoi, Viêt-nam

RAJOSON L. (1988), *Analyse écologique des conditions et des contraintes dans l'étude des phénomènes de transposition didactique : trois études de cas*, Thèse en Didactique des Mathématiques, Université d'Aix-Marseille II

RAMUNNI J. (1989), *La physique du calcul - Histoire de l'ordinateur*, Collection « Histoire et Philosophie des Sciences », Edition Hachette

RABARDEL P. (1995), *Les hommes et les technologies – Approche cognitive des instruments contemporains,* Edition Armand Colin

RAVEL L. (2003), *Des programmes à la classe : étude de la transposition didactique interne – Exemple de l'arithmétique en terminal S spécialité en Terminal S spécialité mathématique*, Thèse en Didactique des Mathématiques, Université Joseph Fourier, Grenoble 1

ROGALSKI J. (1984), « Enseignement de l'informatique en Seconde : une expérience » in *Séminaire de Didactique des mathématiques et de l'informatique*, N° 48-59, Imag (pp. 60-80)

ROGALSKI J. (1985), « Alphabétisation informatique » in *Bulletin APMEP*, N° 347 (pp. 61-74)

ROGALSKI J. (1988), « Acquisition de structures conditionnelles : effet des pré requis logiques et des représentations du dispositif informatique » in *Annales de Didactique et de Sciences cognitives, Vol. 1* (pp. 131-152)

ROSSI (2003), *Histoire de l'informatique* sur http://histoire.info.online.fr/

ROUCHIER A. et *al*, (1987), « Didactique de l'informatique » in *Didactique et acquisition des connaissances scientifiques*, (pp. 341-360), Edition La pensée Sauvage

SAMURÇAY R. (1985), « Signification et fonctionnement du concept de variable informatique chez des élèves débutants » in *Educationnal Studies in Mathematic* N° 16-2 (pp. 143-161)

SAMUÇAY R., ROUCHIER A. (1985), « De ''faire'' à ''faire faire'' : planification d'action dans la situation de programmation » in *Enfance*, N° 2-3, (pp. 241-254)

SATYAM (2005), *Babbage Difference Engine* sur http://www.satyam.com.ar/Babbage/

SIGNAC L. (2004), *Algorithmique et programmation*, Cours en 1ère année à l'Ecole Supérieure d'ingénieurs de Poitiers

SWADE D. (1993), « Le calculateur mécanique de Charles Babbage » in *Pour la Science* N° 186, (pp. 78-84)

TRAU P. (2005), *Cours d'Informatique* sur http://www-ipst.u-strasbg.fr/pat/program/

TROUCHE L. (1996), *Etudes des rapports entre processus de conceptualisation et processus d'instrumentation*, Thèse en Didactique des Mathématiques, Université de Montpellier II

TUCKER A-B. (1986), *Les langages de programmation*, Edition Mc Graw - Hill

VEIGNEAU S. (1999), *Approches impérative et fonctionnelle de l'algorithmique - Applications en C et en Caml light*, Edition Springer

VERILLON P. (1996), « La problèmatique de l'instrument : un cadre pour penser l'enseignement du graphisme » in *Revue Graf &Tec, N° 0,* Université Fédérale Santa Catarina, Brésil

VERROUST G. (2004), *Histoire, épistémologie de l'informatique et révolution technologique* sur http://hypermedia.univ-paris8.fr/Verroust/cours/

VOLLE M. (2003), *Du côté de l'ordinateur* sur http://www.volle.com/

VON NEUMANN J. (1945), *First Draft of a Report on the EDVAC*, Moore School, University of Pennsylvania

WALKER J. (2003), *The Analytical Engine* sur http://www.foumilab.ch/babbage

Dictionnaire Encylopaedia, tomes 1, 12, 19

Rapports de commission Kahane (2001), Dacunha-Castelle (1989)

Programmes scolaires

1. Programmes des mathématiques au lycée en France des années 70, 80, 90, 2000
2. Programmes des mathématiques au lycée au Viêt-nam des années 1990, 2000 et programmes en expérimentation 2002
3. Programme en expérimentation de l'informatique de 1994 - 1995 à 1997 - 1998

Manuels scolaires :

1. Collection *Math* 2^{e}, 1^{er} S, Edition Belin 2000
2. Collection *Déclic Maths* 2^{e}, 1^{er} S, T^{le} S, Edition Hachette 2001
3. Collection *Mathématiques* 2^{e}, 1^{er} S, T^{le} S, Edition Bréal 2002
4. Manuels de mathématiques au lycée *Đại số 10*, *Đại số và giải tích 11*, *Giải tích 12*, Nhà xuất bản Giáo dục Hà nội Việt nam 1998
5. Trois collections de manuels *Đại số 10* (Mathématiques en Seconde), Nhà xuất bản Giáo dục Hà nội Việt nam 1990

6. Manuels d'informatique au lycée (le programme d'expérimentation) *Tin học 10*, *Tin học 11*, *Tin học 12* (Informatique en Seconde, en Première et en Terminale), Nhà xuất bản Giáo dục Hà nội, Việt nam 1994

Sites Internet

http://aconit.org/expositions/histoire_memoire/index.php/
http://febcm.club.fr/index.htm
http://histoiredechiffres.neuf.fr/compter/pascaline.htm
http://histoire-informatique.org/grandes_dates/1_1.html#pmc
http://perso.wanadoo.fr/therese.eveilleau/pages/truc_mat/textes/pascaline.htm
http://q-basic.xodox.de/ALGOL
http://valvassori.free.fr/p_unix.php3
http://www.cl.cam.ac.uk/UoCCL/misc/EDSAC99/simulators/india/
http://www.wikipedia.org/wiki/Algol_%28langage%29
http://www.commentcamarche.net/asm/assembleur.php3
http://www.miralab.unige.ch/subpages/outilsinformatiques/cours/langagebinaire.pps#261,3,Exemple 1

Annexes

Table des matières

Annexe du chapitre A3

Tentative d'un enseignement d'informatique secondaire au Vietnam

Introduction

Une expérimentation d'un enseignement informatique secondaire (EIS) a eu lieu à partir de l'année scolaire 1995-1996, année où commence la mise place de 3 options : sciences naturelles, sciences naturelles et technologiques, sciences sociales dans l'enseignement secondaire. Cet enseignement concerne donc seulement des élèves dont le lycée est choisi pour l'expérimentation. Cette expérimentation a duré pendant 3 ans.

I. Analyse du programme

I.1. Objectifs et organisation de l'enseignement

Le programme précise que l'introduction d'informatique dans l'enseignement secondaire peut se faire suivant trois approches suivantes :

- Introduire l'informatique comme une discipline nouvelle dans l'enseignement secondaire ;
- Présenter des applications de l'informatique (l'ordinateur) à savoir des différents logiciels comme un outil efficace de la nouvelle technologie dans l'enseignement ;
- Utiliser l'ordinateur comme un outil pour un travail professionnel : l'école doit fournir à l'élève des savoirs faire fondamentaux dans l'utilisation de l'ordinateur dans différents champs de travail comme la bureautique, la comptabilité, la conception, le management etc.

Dans la première approche, l'informatique est une discipline commune aux élèves des les lycées à option. Le programme vise à fournir à l'élève une culture informatique dans les différents domaines suivants :

- Les connaissances générales sur l'informatique et sur l'ordinateur ;
- Des notions sur l'algorithme et sur la programmation afin de résoudre des problèmes simples ;
- L'utilisation de l'ordinateur (le système MS-DOS) et de quelques logiciels utilitaires : traitement de texte, gestion d'une base de données, tableur, système de conception automatique, etc.

L'introduction de ces contenus a pour but de :

> […] Contribuer au développement de la pensée de l'élève d'une manière complète tout en se centrant sur la pensée algorithmique afin d'aider l'élève à être capable de considérer et d'organiser un processus d'exécution d'un travail demandé (en prenant en compte la faisabilité et l'efficacité du processus). En résumé, l'introduction d'informatique dans l'enseignement secondaire et l'apprentissage d'autres disciplines aide l'élève à être compétent dans l'évaluation, l'organisation et le traitement de l'information ;
> - Grâce à l'informatique, l'élève doit s'habituer à travailler en suivant un plan ou un programme prédéfini. Ceci est aussi le caractère important de la pensée algorithmique […]

Cette introduction est structurée autour des thèmes « Informatique générale ; algorithme et programmation ; logiciels utilitaires ».

Ces contenus sont répartis dans les modules :

Informatique générale (IG) : 34 séances dont 30 de cours, 2 de révision, 2 de contrôle (34s : 30C-E + 2R + 2Co)
- IG1 : Connaissances générales sur l'informatique et l'ordinateur (14s : 10C + 2E + 1R + 1Co) ;
- IG2 : Système d'exploitation MS-DOS (10s : 4C + 2E + 4TP) ;
- IG3 : Traitement de texte (8s : 4C + 4TP) ;

Langage de programmation (LP)
- LP1 : Bases sur le langage Pascal (32s : 12C + 8 E + 8 TP + 2R + 2Co) ;
- LP2 : Connaissances plus élevées sur le langage Pascal (32s : 12C + 10 E + 6 TP + 2R + 2Co) ;

Gestion de données
- F1 : Bases sur le logiciel Foxbase (32s : 16 C + 12 TP + 2R + 2Co) ;
- F2 : Connaissances plus élevées sur le logiciel (32s : 12 C + 8 E + 8 TP + 2R + 2Co) ;

Logiciels utilitaires
- T : Tableurs (33s) ;
- C : Conception automatique (33s) ;

Voici la répartition des modules pour les 3 options existantes à l'époque. L'enseignement est dispensé à raison de 2 séances par semaine en classe de 10e et 1 séance par semaine en classes de 11e et de 12e.

Classes	Section sciences naturelles	Section sciences naturelles et technologie	Section sciences sociales
10e	IG + LP1	IG + LP1	IG + F1
11e	LP2	LP2 ou F1 ou C	F2 ou P1
12e	T	T	T

L'introduction de la notion d'algorithme est abordée dans la partie I « Informatique générale », chapitre I, intitulé « Connaissances générales sur l'informatique et l'ordinateur » qui est organisé comme suit :

Contenus	**Contenus**
§ 1. Information et traitement d'information § 2. Ordinateur § 3. Représentation de l'information dans l'ordinateur	§ 4. Algorithme § 5. Langage de programmation. Système d'exploitation

L'enseignement de programmation est introduit dans la partie II « Connaissances de base », intitulée « Pascal » dont nous décrivons les contenus comme suit :

Contenus	**Contenus**
§ 1. Introduction § 2. Travail dans l'environnement du Turbo TP § 3. Programmation des calculs numériques § 4. Systématisation des éléments de base du TP § 5. Des types de données simples	§ 6. Instructions de branchement et blocs § 7. Instruction WHILE § 8. Instruction FOR § 9. Instruction de REPEAT § 10. Chaîne de caractères § 11. Tableau

Comme nous l'avons vu, l'introduction de l'informatique et de la programmation est faite dès la classe de 10e. Nous allons donc analyser l'introduction de la notion d'algorithme dans la partie IG1 et puis de la programmation en Pascal dans la partie LP1.

I.2. Enseignement de la notion d'algorithme

L'analyse du programme nous permet de dégager les quelques points essentiels suivants :

I.2.1. Importance et transversalité de la notion d'algorithme

Le programme insiste sur l'importance de la notion d'algorithme et de la pensée algorithmique :

> L'enseignement devrait pleinement être conscient que les connaissances sur la notion d'algorithme et la pensée algorithmique sont parmi des connaissances fondamentales de la culture algorithmique de l'enseignement secondaire. On aura l'occasion de présenter cette notion d'une manière détaillée dans l'apprentissage d'un langage de programmation.

Cette notion sera aussi révisée dans l'enseignement de la programmation comme le précise le programme :

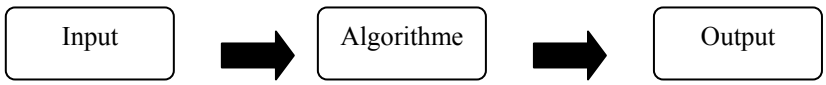
> L'algorithme est une notion très importante et est abordée d'une manière explicite ou implicite dans plusieurs parties dans les manuels d'informatique du lycée.

I.2.2. Algorithme et problème

La relation entre l'algorithme et la résolution d'un problème est soulignée :

> Dans une certaine mesure, on peut considérer que le développement de la mathématique est lié étroitement au processus de résolution des problèmes. […] Le mot problème ici désigne une classe des problèmes du même type.[…] Un problème comporte deux composants essentiels :
> Information entrée (input) : informant sur ce qui est donné ;
> Information sortie (output) : informant sur ce qui est à trouver à partir de l'input ;
> Différente de la résolution d'un problème dans le sens mathématique traditionnel, la résolution d'un problème en informatique consiste à exécuter une série d'actions élémentaires pour arriver d'une manière explicite et claire à l'output à partir de l'input. En principe, ces actions peuvent être confiées à l'ordinateur. Cet ensemble de règles est appelé algorithme du problème donné.

On peut alors schématiser le lien entre input, output d'un problème et son algorithme comme suit :



A noter qu'à aucun endroit le programme ne mentionne le travail qui peut exister dans la comparaison des algorithmes ni la validité d'un algorithme. Autrement dit, ces éléments relevant de l'algorithmique ne semblent pas être dans les intentions des auteurs du programme.
Analysons maintenant le programme pour l'enseignement de programmation.

I.2.3. Absence d'un langage algorithmique structuré

Parmi les 3 manières de représenter un algorithme : organigramme, description algorithmique avec numérotation de ligne, description dans un langage algorithmique structuré, le programme précise :

> Il est noter que pour assurer le caractère systémique, le manuel aborde 3 manières pour représenter un algorithme. Mais il est important d'expliquer les deux premières. Quant au langage avec des structures de contrôle, il suffit de donner un exemple car cette manière sera étudiée d'une manière plus détaillée dans le chapitre concernant le langage de programmation. A présent, il n'est pas exigé de l'élève d'avoir à comprendre complètement cette manière.

Ainsi nous pouvons conclure que le programme semble vouloir s'appuyer sur l'enseignement de la programmation pour introduire des éléments de la notion d'algorithme.

I.3. Enseignement de la programmation

Quels sont les objectifs de cette introduction de l'enseignement du langage Pascal :

> L'enseignement de la « programmation de base » a pour but de fournir aux élèves des connaissances concernant :
> - la programmation des problèmes simples en TP, éléments dans un programme ; relations entre ces éléments et relation entre algorithme et programme ;
> - quelques connaissances sur le langage TP : éléments du langage ; l'écriture d'un programme, le compiler et l'exécuter sur l'ordinateur ;

I.3.1. Choix du langage

Pourquoi le langage TP a-t-il été choisi comme langage de programmation dans l'enseignement ? Le programme avance quelques raisons :

> -Etant un langage de programmation polyvalent structuré, le langage TP a été choisi parmi quelques langages structurés dans l'enseignement de la programmation. […]
> - D'une part il est facile d'implémenter TP […] d'autre part ce langage offre au programmeur plusieurs avantages dans la résolution des problèmes.

Ce choix s'effectue alors sur le caractère structuré du langage et la facilité de son implémentation. Quant aux avantages que ce langage peut offrir à l'usager, le programme reste vague et imprécis.

I.3.2. Formulation des étapes dans la résolution d'un problème sur l'ordinateur

Les étapes à suivre lors de la résolution d'un problème sur l'ordinateur sont précisées :

- Poser le problème : déterminer les données d'entrée et données de sortie (les résultats à trouver) et les relations entre ces données du problème ;
- Choisir l'algorithme ;
- Programmer cet algorithme dans un langage de programmation ;
- Tester le programme avec des données de teste ;
- Exécuter le programme avec des données réelles.

Est mise en avant la méthodologie de la programmation d'un problème. Mais on peut remarquer de nouveau l'absence d'attention sur les éléments d'algorithmique qui permettent un contrôle des programmes une fois que ceux-ci sont écrits.
Comment ces étapes sont-elles mises en place par des manuels ?

II. Analyse des manuels

Comme dans l'analyse du programme nous centrons notre analyse sur les notions de variable et de boucle.

II.1. Partie « cours » de l'introduction à la notion d'algorithme

II.1.1. Exposé sur la notion d'algorithme

L'introduction de la notion d'algorithme est faite après celle des connaissances concernant l'information et le traitement de l'information, l'architecture d'un ordinateur et la représentation des donnés dans l'ordinateur. Cette rubrique comporte trois points :

- Notion d'algorithme ; exemples ;
- Caractéristiques d'un algorithme ;
- Représentation d'un algorithme ;

La définition de cette notion est liée à l'entrée et à la sortie d'un problème :

> Etant donné un problème. L'algorithme de ce problème est un système cohérent et clair de règles visant à déterminer une suite d'actions sur les données d'entrée (input), de telle sorte qu'après un nombre fini d'exécution de ces actions on obtient le résultat (output) du problème. (p. 44)

L'exemple suivi est un algorithme itératif, celui d'Euclide représenté en langage algorithmique avec la numérotation de ligne.
La notion de variable et la notation d'affectation sont évoquées :

> Si r ≠ 0 alors affecter b à a, r à b. Retourner à l'étape 1 […] ;
> L'affectation : attribuer une valeur concrète à une variable ; (p. 45)

Le tableau donné par la suite simule l'exécution de cet algorithme pour deux nombres a = 68 et b = 119 et permet ainsi de suivre le changement de valeur des variables a, b et r.
Les caractéristiques d'un algorithme « finitude, déterminisme, donnée d'entrée (input), donnée de sortie (output), efficacité (en termes de mémoire, de temps), universalité » sont introduites.

II.1.2. Deux types de langage : description séquentielle et organigramme

Le manuel présente aussi 3 manières de représenter des algorithmes : description séquentielle avec numérotation de ligne, organigramme (avec l'algorithme d'Euclide), langage algorithmique structuré. Alors que deux exemples sur les deux premiers types de langages sont introduits, le manuel ne donne aucun exemple concernant le langage algorithmique en précisant :

> Ce sont les langages de programmation qui sont une forme de représentation des algorithmes (voir §5), dans laquelle chaque structure de contrôle sont exprimer par une instruction correspondante. (p. 51)

Le langage algorithmique structuré est alors assimilé à un langage de programmation.

II.2. Partie « cours » de l'introduction à la programmation

II.2.1. Structure de cours comme celle au niveau universitaire

La répartition des contenus de cours est la suivante :

Contenus	Contenus
§ 1. Introduction 1. Un programme TP simple 2. Démarrer le logiciel TP 3. Traitement de texte dans TP **§ 2. Travail dans l'environnement du Turbo TP** 1. Environnement du TP 2. Travailler avec le menu **§ 3. Programmation des calculs** 1. Constante et variable 2. Représentation d'un nombre réel 3. Les opérations 4. Instruction d'affectation 5. Entrée/Sortie des données (4C + 2TP pour §1, §2 et §3) **§ 4. Systématisation des éléments de base du TP** 1. Table de lettres utilisées dans un programme 2. Mots clés et des accolades 3. Noms de standard 4. Constant textuel 5. Explication d'un programme **§5. Des types de données simples** 1. Notions dur type de donnée 2. Type entier 3. Type réel 4. Type caractériel 5. Type logique (2C + 2E pour §4 et §5)	**§ 6. Instructions de branchement et composées** 1. Instruction de branchement incomplète 2. Instruction de branchement complète 3. Expressions d'une condition 4. Instructions composées **§7. Instruction WHILE** 1. Problème 2. Algorithme 3. Instruction itérative avec condition avant 4. Programme 5. Exemples (2C + 2E + 2TP pour §6 et §7) **§8. Instruction REPEAT** 1. Problème 2. Instruction itérative avec condition après 3. Programme 4. Exemples **§9. Instruction FOR** 1. Problème 2. Algorithme 3. Instruction itérative avec condition avant 4. Programme 5. Exemples (2C + 2E + 2TP pour §8 et §9) **§10. Tableau** **§10. Chaîne de caractères** (2C + 2E + 2TP pour §10 et §11)

La structure du cours est calquée sur celle du programme de l'enseignement de programmation au niveau universitaire donné par MEF (cf. Annexe B1).

II.2.2. Définition de la notion de variable sans relation avec la notion de mémoire

Cette définition est introduite à l'occasion de la présentation d'un programme de calcul (rubrique § 3) sans faire allusion à la notion de mémoire :

> Des variables sont des quantités dont les valeurs peuvent changer durant l'exécution du programme. Les variables ont des noms afin de se distinguer. (p. 131)
> Instruction affectation réalise le calcul de valeur d'une expression puis « affecter » la valeur trouvée à une variable. Ceci s'écrit : <variable> := <expression>. La notation := est appelé affectation. (p. 127)

La partie exercices de cette rubrique n'introduit aucun exercice portant sur cette notion ou sur la notation d'affectation.

II.2.3. Absence de langage algorithmique structuré et langage des organigrammes

Le problème suivant sert à introduire dans cet ordre les boucles « While » et « Repeat » :

> Le taux d'intérêt pour une épargne à durée indéterminée est k %. Une personne met dans son livret une somme initiale a. Après combien de temps cette personne obtient-elle une somme qui n'est pas plus petite que b ? (p. 148)

La boucle « While » est introduite comme suit :

Algorithme
La somme au début du mois est a. A la fin du mois, la somme comprend la somme initiale plus l'intérêt (a + a * k). C'est aussi la somme mise dans le livret pour la prochaine mois. Ici, on ne sait pas combien de fois on a à répéter ce calcul. Le processus d'itération sera fini dès que la somme obtenue ne sera pas plus petite que b. L'instruction itérative avec condition est outil de TP capable d'exprimer l'itération mentionnée. (p. 149)

L'explicitation de l'algorithme du problème consiste alors à trouver la somme d'argent au bout d'un mois. La présentation en langage d'algorithmique structuré n'est pas abordée. La présence de langage algorithmique n'est introduite dans aucun autre endroit du manuel.

II.2.4. Absence d'introduction d'éléments d'algorithmique

Le travail sur les variables présentes dans le problème : a, b, k et celle à introduire : t n'est pas explicité. Ces variables sont introduites en même temps que le programme est donné ;
Nous résumons ici les trois programmes donnés par le manuel utilisant 3 types de boucles (nous gardons seulement les instructions faisant partie de la boucle) :

Boucle « While »	Boucle « Repeat »

BEGIN
t := 0 ;
WHILE a < b DO
  BEGIN
  a := a + a * k ;
  t : = t + 1;
  END;
WRITE ('A y mettre dans ',t,' mois);
END
	BEGIN
	t := 0 ;
	REPEAT
	  BEGIN
	  a := a + a * k ;
	  t : = t + 1;
	  UNTIL a > = b ;
	WRITE ('A y mettre dans ',t,' mois);
	END

- Le travail mathématique sur une autre façon de calculer la somme au bout de t mois : $a*(1 + k)^t$ n'est pas explicité. On peut en effet obtenir 2 autres programmes :

Boucle « While »	Boucle « Repeat »

BEGIN
t := 0 ;
WHILE a *(1 + k) ^t < b DO
  BEGIN
  t : = t + 1;
  END;
WRITE ('A y mettre dans ',t,' mois);
END
	BEGIN
	t := 0 ;
	REPEAT
	  BEGIN
	  t : = t + 1;
	  UNTIL a *(1 + k) ^t > = b ;
	WRITE ('A y mettre dans ',t,' mois);
	END

Pour la boucle « For », la question est « calculer la somme d'argent après t mois ». Si on utilise la formule a * (1 + k)t, la boucle « For » n'est plus nécessaire.

Boucle « For »	Sans boucle
BEGIN WRITE ('Donner le nombre de mois') ; READLN (t) ; FOR i := 1 TO t DO a := a + a * k ; WRITE ('La somme après ',t,' mois :', a: 9 :2) ; END	BEGIN WRITE ('Donner le nombre de mois') ; READLN (t) ; WRITE ('La somme après ',t,' mois :', a *(1 + k) ^t : 9 :2) ; END

II.3. Partie « exercices » de l'introduction à la notion d'algorithme

II.3.1. Absence des algorithmes itératifs

Sauf l'algorithme d'Euclide donné comme seul l'exemple dans le cours, les 9 algorithmes introduits dans la partie exercices sont non itératifs et 6 d'entre eux sont issus des problèmes mathématiques dans l'EMS.
A ce propos ces exercices sont répartis ainsi :
T1 : Ecrire un algorithme pour un problème donné (3) ;
T4 : Expliquer un algorithme (2) ;

Sur trois algorithmes à écrire, deux sont ceux en mathématiques dont la solution ne devra pas poser des problèmes. Par exemple :

> Sachant les mesures de l'angle B et du côté BC, écrire un algorithme pour calculer les mesures de l'angle C et les côtés AC, AB. (p. 52)

II.3.2. Absence de la notion de variable

Comme on peut le constater, aucun exercice ne porte sur la notion de variable.

II.4. Partie « exercices » de l'introduction de la programmation

II.4.1.Concernant les types de boucles

Les types de tâches des exercices concernant 3 boucles sont donnés dans le tableau suivant :

Type de tâches	**Boucle « While »**	**Boucle « Repeat »**	**Boucle « For »**	**Total**
T1	3	1	11	25
T3	1	1	1	3
Total	4	2	12	28

Rappelons les types de tâches T1 et T3 :
T1 : Ecrire un algorithme ou un programme informatique pour un problème donné ;
T3 : Trouver le résultat d'un algorithme donné pour différentes instances d'un problème ;
Les exercices ne sont pas nombreux. On peut même noter un manque pour la mise en œuvre des boucles.
C'est dans la rubrique concernant la boucle « For » qu'il y a le plus d'exercices. Il s'agit des exercices mathématiques ou d'autres ?

Type d'exercices	Boucle « While »	Boucle « Repeat »	Boucle « For »	Total
Mathématiques	3	0	11	14
Non maths.	1	2	1	4

Analysons d'une manière plus détaillée les techniques et technologies à ces types de tâches repérés dans ces exercices. Nous ne tenons compte que des exercices en mathématiques.

Type de tâche T1

Voici les exercices proposés avec des boucles :

Boucle « While »

+) Ex.1 « Ecrire un programme pour calculer la somme d'une suite de nombres entrés depuis le clavier. La condition d'arrêt est faite quand le nombre égale à zéro ».
+) Ex2. « Calculer la somme S = 1 + 1/2 + 1/3 + … Le processus de calcul s'arrête quand 2 – S < 0.001. »

Boucle « For »

+) Ex1. « Ecrire un programme afin que l'ordinateur imprime la racine carrée de tous les nombres paire de 0 à 50 ».
+) Ex2. « Ecrire un programme afin que l'ordinateur annonce le quotient et le reste dans une division euclidienne d'un nombre a par 5 (a est entré depuis le clavier) ».
+) Ex3. « Dresser les valeurs des fonctions suivantes :
a) $y = x^2 + 2x + 1$ avec x un nombre entier appartenant à [- 2 ; 5] ;
b) $y = \sqrt{x}$ avec x entier un nombre entier appartenant à [1, 5] ; ».
+) Ex4. « Déterminer les solutions qui sont des nombres entiers positifs de l'équation 2x + 4y = 100 ».
+) Ex5. « Ecrire un programme afin que l'ordinateur imprime à l'écran les naturels ayant 3 chiffres sachant que le reste de la division de ces nombres par 6 est 2 et la somme des ses chiffres est 20 ».
+) Ex6. « Ecrire un programme afin que l'ordinateur imprime à l'écran les naturels ayant 4 chiffres sachant que la somme des ses chiffres des chiffres des milliers et des centaines est égale à celle des chiffres des dizaines et des unités ».
+) Ex7. « Ecrire un programme résolvant le problème suivant : déterminer le nombre de chiens et de poulets sachant qu'ils sont 36 et le nombre total de leurs pattes est 100 ».

+) Ex8(*)[1]. « Ecrire un programme afin que l'ordinateur imprime à l'écran la table de multiplication des nombres de 0 à 9 ».
+) Ex9(*). Déterminant tous les nombres plus petit que 1000 sachant que chacun est égal à ses divisibles (sauf lui-même) ».
+) Ex10. « Ecrire un programme résolvant le problème suivant : déterminer le nombre de types de buffles sachant qu'ils sont 100 et qu'il y a 100 paquets de foins répartis tel que les buffles debout en mangent 5 par chacun, les buffles allongés en mangent 3 par chacun et tous les 3 vieux buffles en mangent 1 ».
+) Ex11(*). « Déterminer tous les naturels a, b ($1 \leq a, b \leq 100$) de telle sorte que $a^2 + b^2$ est le carré d'un nombre ».

Cette liste appelle les premières remarques suivantes :

Des problèmes mathématiques à dominant arithmétique au niveau collège (programme vietnamien)

Les problèmes mathématiques sont au niveau du collège dans le programme vietnamien. Ils travaillent essentiellement sur les nombres naturels ou entiers relatifs. On peut noter la présence de plusieurs exercices concernant l'équation de Diophante : existe-t-il la solution (x, y,…) à une équation diophantienne D(x, y,…) = 0.

L'établissement de la condition d'arrêt d'une boucle est donné dans l'énoncé

En effet, dans les deux exercices avec la boucle « While », la condition d'arrêt est établie dans l'énoncé alors que dans ceux avec la boucle « For », la condition est explicitée d'emblée suivant la formulation du problème. Cela ne veut pas dire que c'est la condition optimale de la solution en terme de temps et de mémoire. Mais c'est seulement la condition d'arrêt qui permet le programme de fonctionner correctement.
Considérons par exemple la solution représentée en langage algorithmique de l'exemple 4. A noter que cet exemple demande la mise en place de deux boucles imbriquées :

Boucle avec condition d'arrêt « donné »

```
Début
Pour i := 1 à 100 faire
   Pour j := 1 à 100 faire
      Si 2 × i + 4 × j = 100 alors
         Afficher i, j
      FinSi
   FinPour
FinPour
Fin.
```

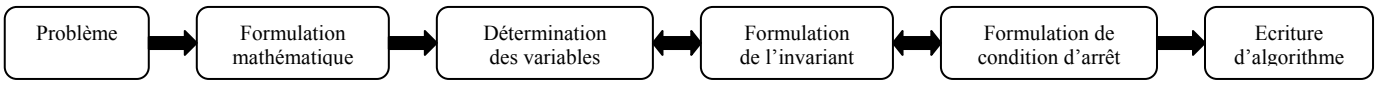
Boucle avec condition d'arrêt « travaillé »

```
Début
Pour i := 1 à 50 faire
   Pour j := 1 à 25 faire
      Si  i + 2 × j = 50 alors
         Afficher i, j
      FinSi
   FinPour
FinPour
Fin.
```

Ici le travail sur la condition d'arrêt consiste à déterminer le plus petit intervalle pour i et j. Ce travail réduit le nombre de répétitions de 10000 à 1250.

Techniques pour le type de tâche T1

Il n'y a pas une technique générale pour ce type de tâche, c'est-à-dire la technique générale pour écrire un algorithme d'un problème donné. Pour faciliter l'analyse, nous proposons les étapes dans l'écriture d'un algorithme aux exercices repérés précédemment :



Les étapes dans l'écriture un algorithme

Ces étapes sont illustrées dans la résolution de ces exercices dans le tableau suivant :

Ex.	Type de tâches	Algorithme
Ex1.	Calculer la somme d'une suite de nombres entrés depuis le clavier. La condition d'arrêt est faite quand le nombre égale à zéro	Début $0 \rightarrow S$ Lire r Tant que $r \neq 0$

[1] Les exercices avec le signe (*) sont considérés par le manuel comme facultatifs.

		S + r → S Fin tant que Fin.
E2.		Début 0 → S, 1 → i Tant que 2 – S ≥ 0.001 S + 1/i → S i + 1 → i Fin tant que Fin.
E3a.	Images de la fonction $f(x) = x^2 + 2x + 1$ avec x ∈ Z dans [-2,5]	Début Pour i de 1 à 50 faire afficher √i Fin pour Fin.
E3b.	Images de la fonction f(x) = √x x ∈ Z dans [1,5]	Début Pour i de - 2 à 5 faire afficher $i^2 + 2i + 1$ Fin pour Fin.
E4.	Déterminer des nombres naturels de l'équation 2x + 4y = 100	Début Pour i de 1 à 100 faire Pour j de 1 à 100 faire Si 2i + 4j = 100 alors Afficher i, j Fin si Fin pour Fin pour Fin.
E5.	Déterminer $\overline{ijk}$ tel que $\overline{ijk}$ = 6n + 2 et i + j + k = 20	Début Pour i de 1 à 9 faire Pour j de 0 à 9 faire Pour k de 0 à 9 faire Si (100i + 10j + k) mod 6 = 2 et i + j + k = 20 alors Afficher 100i + 10j + k Fin si Fin pour Fin pour Fin pour Fin.
Ex6.	Déterminer $\overline{ijkl}$ tel que i + j = k + l	Début Pour i de 1 à 9 faire Pour j de 0 à 9 faire Pour k de 0 à 9 faire Pour l de 0 à 9 faire Si i + k = j + l alors Afficher 1000i + 100j +10 k + l Fin si Fin pour Fin pour Fin pour Fin pour Fin.
Ex7.	Déterminer x, y naturels tels que $\begin{cases} i + j = 36 \\ 2i + 4j = 100 \end{cases}$	Début Pour i de 1 à 36 faire Pour j de 1 à 36 faire Si i + 2j = 100 alors Afficher i, j

		Fin si Fin pour Fin pour Fin.
Ex8.	Table de multiplication ij avec i, j ∈ [0, 9]	Début Pour i de 0 à 9 faire Pour j de 0 à 9 faire Afficher i j Fin pour Fin pour Fin.
Ex9.	Déterminer tous les nombres inférieurs à 1000 tels que chacun soit égale à la somme de ses divisibles	Début Variable s Pour i de 1 à 1000 faire 0 → S Pour j de 1 à j - 1 faire Si mod (i,j) = 0 alors S + j → S Fin si Fin pour Si S = i alors Afficher i Fin si Fin pour Fin.
Ex10.	Déterminer i, j, k naturels tels que : $\begin{cases} 5i + 3j + k / 3 = 100 \\ i + j + k = 100 \end{cases}$	Début Pour i de 1 à 100 faire Pour j de 1 à 100 faire Pour k de 1 à 100 faire Si 5i + 3j + k/3 = 100 et i + j + k = 100 alors afficher i, j, k Fin si Fin pour Fin pour Fin pour Fin.
Ex11.	Déterminer i, j ∈ [1, 100] tels que $i^2 + j^2 = k^2$ avec k < 20000	Début Pour i de 1 à 100 faire Pour j de 1 à 100 faire Pour k de 1 à 10000 Si $i^2 + j^2 = k^2$ alors afficher i, j Fin si Fin pour Fin pour Fin.

Quasi absence de problèmes sur la notation d'affectation donc sur la notion de variable

Pour la boucle « For », la condition d'arrêt est presque donnée et la mise à jour de la variable compteur se fait automatiquement par l'instruction « pour i de n_1 à n_2, pas *p* faire …» et le nombre d'exercices sur la boucle « While » dans laquelle il est question de la création des variables dans la boucle ainsi que la mise à jour de ces variable, est modeste donc on peut conclure sur la quasi-absence d'un travail sur la notion de variable dans une boucle.

Quasi absence de la formulation de l'invariant de la boucle

Sauf quelques problèmes où l'invariant de la boucle diffère de la formulation mathématique du problème, on peut noter que l'invariant de la boucle est donné dès que la formulation mathématique est faite. Or cette formulation est presque donnée dans l'énoncé.

Annexe du chapitre B1

Etat actuel de l'introduction de l'enseignement informatique au niveau universitaire en France et au Viêt-nam

Introduction

L'analyse produite ici s'appuie sur le questionnement suivant :
Quel(s) langage(s) algorithmique(s) ? Quel(s) langage(s) de programmation ? Quelles organisations informatiques sur boucle et variable ?
Les résultats de l'étude historique conduite dans le chapitre B1 sur la naissance et l'évolution d'un enseignement universitaire d'algorithmique et de programmation nous conduisent à affiner ce questionnement : quels sont les différents choix d'enseignement ? quelles sont les raisons données à l'enseignement d'un « langage algorithmique » dans ces différents choix d'enseignement ? quelles relations a ce langage avec les différents langages de programmation ? quels statuts ont les objets d'enseignement « boucles » et variables », invariants actuels dans les enseignements au début de l'Université ?

I. Etat au Vietnam

I.1. Quelques généralités et précisions sur le programme de l'informatique générale

Dans cette partie nous nous posons des questions suivantes :
- Où l'informatique est-elle enseignée ?
- Quels sont les savoirs enseignés ?
- Dans l'introduction informatique, quels sont les objets d'enseignement ?
- Comment les deux parties : langage de programmation et algorithmique sont-elles présentées ?
- Quelle est organisation didactique et informatique des objets d'informatique : machine, boucle, variable ?

L'enseignement informatique (basic Informatics – terme traduit par le ministère de l'éducation et de la formation MEF du Vietnam) de base (ou Informatique générale) est dispersé sur 3 départements d'université qui sont :
- Mathématique (dans des écoles normales supérieures, 3 universités nationales et quelques grandes écoles) ;
- Mathématiques appliquées et informatique (dans 3 universités nationales et quelques grandes écoles) ;
- Technologie d'information[1] (qui existe dans plusieurs universités au Vietnam) ;

Dans chacun de ces trois départements, en 1ère année, l'enseignement de l'informatique de base donne 4 à 6 crédits sur les 62 à 67 crédits des unités d'enseignement général pour les étudiants en mathématiques ou mathématiques appliquées ou 8 (sur 55) crédits pour les étudiants en technologie l'information.
L'examen des programmes de ces départements de l'université nationale et quelques autres universités montre que :

[1] Dans le décret 49/CP du gouvernement du Vietnam en 1996, le terme « technologie de l'information » est défini comme suit : « Technologie de l'information est l'ensemble des méthodes scientifiques, des moyens et outils techniques modernes, essentiellement des techniques de l'ordinateur et de télécommunication, qui vise à organiser, exploiter et utiliser d'une manière efficace des ressources très riches et potentielles d'information dans tous les domaines d'activité de l'homme et de la société… La technologie de l'information est développée sur la base des technologies de l'informatique, de l'électronique, la télécommunication et la cybernétique. » (cité dans Nguyễn V.Q.H 2002, p.6)

- Dans les départements autres que celui de mathématique, à côté du langage Pascal d'autres langages de programmations sont enseignés dans les matières comme techniques de programmation ou programmation approfondie comme C, C++, Visual Basic, Fortran. Ils sont tous du style impératif.
- Les autres matières qui concernent la notion d'algorithme : automate et algorithme, structure des données et algorithme, théorie des graphes et algorithmes, conception et évaluation des algorithmes sont abordées à partir de la fin de la 2e année ou de la 3e année.

Regardons maintenant le programme de l'enseignement de l'informatique de base en 1ère année de l'université au Vietnam. Nous choisissons celui destiné à la formation des étudiants en Mathématique. Rappelons l'un des objectifs de la formation fixés par MEF :

> Le programme vise à entraîner l'étudiant à l'esprit rigoureux de la mathématique, l'esprit d'algorithmique et des méthodes scientifiques pour approcher des problèmes pratiques. (Programme 2002)

Ce programme occupe 4 crédits en 60 séances dont 34 pour cours et exercices et 26 pour travail pratique.

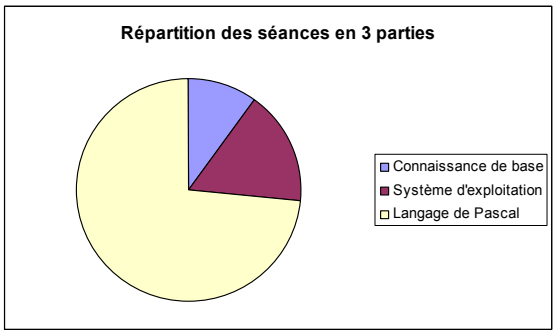
> **Objectifs**
> Fournir aux étudiants des filières des sciences naturelles des connaissances les plus fondamentales de l'Informatique. Les étudiants sauront utiliser aisément l'ordinateur, comprendre le réseau d'ordinateurs afin s'en servir, programmer des problèmes scientifiques en langage Pascal. (Programme 2002)

Les contenus de chacun de ces thèmes sont décrits dans chaque chapitre mentionné ci-dessous.

> *Partie 1. Introduction à l'informatique (cours et exercices : 6 séances)*
> - Chapitre 1. Traitement de l'information sur l'ordinateur : 1.1. Notion de l'information ; 1.2. Codage de l'information ; 1.3. Traitement de l'information ; 1.4. Architecture de l'ordinateur ; 1.5. Principe de Von Neumann ; 1.6. Bases logique et arithmétique de l'ordinateur (système de numération, représentation de l'information dans l'ordinateur, opération logiques et circuits logiques) ;
>
> - Chapitre 2. Langage de l'ordinateur et système d'exploitation : 2.1. Langage machine ; 2.2. Système d'exploitation (ses principaux fonctionnements, ses composants essentiels, organisation par répertoire) ;
>
> - Chapitre 3. Algorithme : 3.1. Algorithmes et caractéristiques d'un algorithme ; 3.2. Méthodes de présenter un algorithme ; 3.3. Quelques algorithmes usuels ;
>
> - Chapitre 4. Langage de programmation et programmes traducteurs : 4.1. Les étapes de résolution d'un problème à l'aide d'ordinateur ; 4.2. Langage de programmation ; 4.3. Programmes traducteurs ;
>
> - Chapitre 5. Généralité sur le réseau d'ordinateur et Internet
>
> *Partie 2. Utilisation de l'ordinateur (cours et exercices 4 séances ; pratique : 6 séances)*
> - Chapitre 6. Système d'exploitation MS-DOS
> - Chapitre 7. Système d'exploitation Windows
>
> *Partie 3. Programmation en langage Pascal (cours et exercices 24 séances ; pratique : 20 séances)*
> - Chapitre 8. Les éléments essentiels du langage Pascal : 8.1. Symboles essentiels ; 8.2. Mots clés ; 8.3. Nom et notation ; 8.4. Constantes textuelles ;
>
> - Chapitre 9. Types de données standardisées : 9.1. Type logique ; 9.2. Type nombres entiers ; 9.3. Type nombres réels ; 9.4. Type caractères ;
>
> - Chapitre 10. Structure d'un programme en langage Pascal : 10.1. Partie déclaration ; 10.2. Partie d'instruction ; 10.3. Environnement intégré et développé du Turbo Pascal 7.0 ;
>
> - Chapitre 11. Les expressions : 11.1. Constantes et variables ; 11.2. Opérations ; 11.3. Appel des fonctions ;
>
> - Chapitre 12. Instructions : 12.1. Instructions simples (affectation, branchement, appel des procédures ; procédures d'entrée et de sortie) ; 12.2. Instructions structurées (instructions composées, instructions conditionnelles, instructions d'itération) ;
>
> - Chapitre 13. Type de données structurées : 13.1. Tableau ; 13.2. Chaîne de caractère ; 13.3. Ensemble ; 13.4. Enregistrement ; 13.5. Fichiers ; 13.6. Fichiers de texte ; (Programme 2002)

Ce programme appelle quelques commentaires :

- A part les objectifs d'enseignement, aucun commentaire n'accompagne ce programme. Ces contenus sont répartis en 3 grands thèmes :
  +) Connaissances de base en Informatique ;
  +) Utilisation de l'ordinateur ;
  +) Programmation ;
- L'algorithmique n'est pas enseignée. C'est plutôt une présentation de la notion d'algorithme : définition, caractéristiques d'un algorithme, quelques algorithmes usuels. Des notions qui relèvent de l'algorithmique comme complexité d'un algorithme, évaluation d'un algorithme, etc. ne sont pas abordées. Or l'un des objectifs de la formation est d'entraîner les étudiants à « l'esprit d'algorithmique » est-ce que cet objectif est pris en charge par l'enseignement de la programmation grâce auquel cet « esprit algorithmique » peut être forgé ? Nous constatons cependant que la présentation de la notion d'algorithme dans un chapitre à part au même titre que d'autres concepts de base en Informatique comme : traitement de l'information, langage de l'ordinateur et système d'exploitation etc.
- La programmation en langage Pascal est un thème important et central dans cet enseignement comme le montre dans l'organigramme suivant.



- Les notions concernant les variables informatiques, par l'instruction d'affectation et boucle, par des instructions structurées d'itération sont présentées dans un chapitre sur les instructions. Notons que l'instruction de branchement est présentée comme une instruction simple et, ce avant les instructions avec boucle. On peut constater que les auteurs soulignent les notions de « constantes » et de « variables » comme constituants importants du programme informatique.

Nous allons analyser quelques livres informatiques. A noter que dans ce programme le MEF met une liste de livre de référence pour cet enseignement de l'Informatique de base dont :

- Hồ Sĩ Đàm (2000), Giáo trình Tin học, Edition « Nhà xuất bản Đại học quốc gia »
- Quách Tuấn Ngọc (1998), Ngôn ngữ lập trình Pascal, Edition « Nhà xuất bản Giáo dục » (Programme 2002)

Nous choisissons les livres de ces deux auteurs (la 1ère édition en 1990 du 1er livre et la 2e pour le 2e en 2002). Nous prenons aussi un autre qui a servi pendant plusieurs années de manuel pour les étudiants en 1ère année de l'université de construction civile de Hanoi :

- Hoàng Nghĩa Tý et Phạm Thiếu Nga (1999), Tin học đại cương, Edition « Nhà xuất bản Khoa học và kỹ thuật »

Dans notre analyse, nous les nommons respectivement M1, M2 et M3.

Nouvelles questions :
- Pourquoi pas d'algorithmique ? Considérations avec la programmation.
- Est-ce que le langage algorithmique est présent dans la programmation ? En quel langage ?
- Pourquoi le langage Pascal ?
- Comment ce programme est-il mis en place dans les livres ? En particulier, boucle et variable ?

La première observation sur M1 et M3 est qu'ils s'organisent autour de 3 grands thèmes définis par le programme du MEF.

Dans les années 90, le système d'exploitation Microsoft n'est pas encore répondu au Vietnam, il est remplacé par MS-DOS.

I.2. Algorithme

Comme nous avons dit, l'algorithmique n'est objet d'enseignement. Nous allons essayer de repérer ce qui concerne la notion l'algorithme[1] dans ces trois manuels, notamment ses caractéristiques et les langages pour décrire un algorithme.
Les définitions de la notion d'algorithme données :

> M1 : L'activité intellectuelle du quotidien est, à vrai dire, le processus de traitement d'information. Ce processus est décrit comme suit : on applique des règles prédéfinies qui permettent, à partir de l'information initiale, de trouver l'information attendue sur le résultat. Ces règles sont déterminées sur la méthode de résolution du problème indiquant les interventions nécessaires et l'ordre dans laquelle ces interventions sont faites. L'algorithme est une description de cette méthode, ces interventions et cet ordre. (p. 9)
> M2 : On peut comprendre que l'algorithme est un ensemble fini des actions (les tâches, les opérations…) qui peuvent être nommées et ces actions sont exécutées dans un ordre déterminé et approprié à certains objets afin d'obtenir la chose souhaitée. L'adjectif « fini » est compris aussi bien dans le sens de temps que dans le sens outils à utiliser. (p. 18)
> M3 : Algorithme est des astuces, des étapes du processus de traitement de l'information sur l'ordinateur. (p. 94)

On constate une diversité dans ces définitions. Seul M1 donne les caractéristiques suivants que doit posséder un algorithme : discret, exact, universel, déterministe, effectif. La caractéristique « fini » est déjà incluse dans M3.
Quant à la façon de présenter des algorithmes, après la définition, M1 donne un exemple d'un algorithme en langage naturel. Dans cet algorithme, les instructions (les auteurs utilisent le terme « opérateur » à la place de « instruction ») sont numérotées. La boucle « si…alors aller à… » est présente. Pourtant c'est le seul exemple. Aucun organigramme n'est présent. On peut comprendre que les auteurs utilisent les programmes en langage pascal pour présenter des algorithmes.
En effet lors d'une classification des langages de programmation, les auteurs du M1 considèrent que les langages algorithmiques sont des langages de programmation évolués.

> **4.3. Les langages algorithmiques**
> Les langages de ce groupe sont universels à un degré élevé et peuvent décrire plusieurs algorithmes différents. Plusieurs éléments de ces langages ont une forme similaire à des symboles, des formules mathématiques usuelles. Dans le domaine d'application, on peut citer : les langages algébriques […] Ces langages apparaissent depuis longtemps : Fortran, Algo, Pascal, PL/1, Basic, Ada… ; les langages dans la gestion […]. (M1, p. 71)

Dans M2, l'auteur présente explicitement 3 méthodes de présenter des algorithmes : en langage naturel grâce à des lignes numérotées et l'instruction « si…aller à… », en langage de programmation et en langage des organigrammes :

> Pascal est un des langages algorithmiques, c'est-à-dire ce langage exprime lui-même l'algorithme à procéder. C'est aussi un des points forts du langage qui aide mentalement le programmeur à avoir la construction de l'algorithme sans avoir besoin l'organigramme. Dans ce livre, l'organigramme des algorithmes est illustré d'une manière concrète, comme un moyen supplémentaire, avec le langage Pascal dans quelques structures d'instructions. (p. 19)

L'organigramme est utilisé donc seulement dans la présentation des structures de contrôle conditionnelle ou itérative. Les programmes sont présentés directement dans le langage Pascal.
A propos de M3, la présentation de différents types d'algorithmes et d'une méthode pour les présenter fait partie d'un chapitre. L'organigramme accompagne presque tous les programmes et le langage algorithmique est assimilé à un langage de programmation, en l'occurrence celui Pascal. La programmation et l'algorithmique ne sont pas distingués. La citation illustre ce propos :

[1] Le mot « algorithme » et « algorithmique » peuvent être traduit en vietnamien en « thuật toán », « thuật toán » ou « giải thuật ». Les auteurs de livres informatiques n'arrivent pas à y avoir un consensus. Il y en a qui pensent que le mot « thuật toán » est utilisé seulement pour désigner des méthodes de résolution en mathématique donc les deux termes « thuật toán » ou « giải thuật » sont plus généraux. Le programme du MEF utilise le mot « thuật toán ». C'est aussi le mot qui s'utilise le plus pour le mot « algorithmique ». Nous le prenons ainsi dans notre analyse.

Dijkstra a proposé une méthode de conception des algorithmes efficace, celle de la programmation structurée. Suivant ce point de vue, les algorithmes sont divisés en structures fondamentales : séquentielle, conditionnelle et itérative. (p. 99)
De même époque qu'Algo, est le langage algorithmique de haut niveau Fortran, en abréviation de Formula Translator construit par Backus. (p. 130)

Cependant dans tous les exemples du chapitre, l'organigramme (avec la notation des instructions comme en langage Pascal, i := i + 1, par exemple) est le seul moyen pour les présenter. Notons que dans un exemple concernant l'algorithme itératif où la notation i := i + 1 est donnée pour la 1ère fois, l'explication est la suivante :

Calculer $y = x^n$ ou $y = \underbrace{x.x.x...x}_{n\ fois}$

On organise comme suit : il y a 4 grandeurs participant au processus de calcul : y, x, n, et un compteur qui compte les étapes à répéter i, au début on affecte x à y, (y := x) ; i prend des valeurs à partir de 2 (i := 2), puis on calcule de manière répétitive y := y*x. Chaque fois qu'on calcule cette formule, on incrémente le compteur i (i := i + 1), ce processus se répète jusqu'à i > n. (p. 105)

Cette notion est aussi présentée dans M1, M2 dans la partie de présentation des organigrammes (avant la partie de programmation) et ce sans explication. On peut interpréter que la notion de variable liée intrinsèquement à cette notion ne pose pas de problème et que on peut le présenter sans aucune manipulation sur la machine (à travers un programme informatique).
En résumé :
- On peut constater une diversité dans la définition de la notion d'algorithme. Il est à souligner que le concept de machine de Turing n'est mentionné dans aucun de ces livres.
- Le langage algorithmique est assimilé à un langage programmation. Dans ce cas, c'est celui de Pascal. Avec l'organigramme ce sont deux méthodes pour présenter un algorithme.
- Les auteurs de M1, M2 soulignent les étapes pour résoudre un problème sur l'ordinateur comme :

M1 : Etape 1. Formuler le problème mathématique ; Etape 2. Choisir une méthode ; Etape 3. Construire un algorithme ; Etape 4. Ecrire l'algorithme dans un langage de programmation ; Etape 5. Corriger le programme ; Etape 6. Exécuter le programme ; (p. 58)
M2 : Problème → Chercher et construire un algorithme → écrire un programme (p. 18)

En réalité les programmes sont toujours donnés sans aucune construction d'algorithme préalable.

I.3. Variable et boucle dans le langage de programmation

I.3.1. Structuration d'un cours de programmation des manuels vietnamiens

Les auteurs de ces livres soulignent tous l'aspect pédagogique et structurel du langage Pascal :

M1 : L'aspect structurel dans Pascal consiste en : définition des données, des opérateurs, des procédures et des fonctions (p. 99) ;
M2 : Ce langage aide les étudiants ainsi que les débutants en programmation à avoir l'habitude d'écrire un programme structuré d'une manière claire, compréhensible et lisible par d'autres personnes ; Pascal est un langage ayant un typage de données très fort ; Pascal est un langage structuré ; (p. 10)
M3 : Wirth, l'auteur du langage Pascal garde toujours son point de vue structuré systématique dans la programmation ainsi que dans l'enseignement. […] Pascal est un langage structuré. Un programme dans ce langage est structuré en blocs, chaque bloc est placé entre le couple des mots clés « begin » et « end ». Ces blocs peuvent être imbriqués ; (p. 132)

Un autre point commun est le recours à un organigramme dans la présentation des boucles dans les trois livres. M3 l'utilise aussi dans presque tous les programmes.
Le tableau de la page suivante montre l'organisation didactique proposée par le programme du MEF et M_i. Comme nous l'avons vu le programme du MEF se présente sous forme en chapitres, nous le mettons au même titre (dans la présentation) que les M_i. Nous respectons les sous chapitres dans M_i jusqu'à l'apparition des notions variable et boucle. Seulement les titres de chapitres qui les succèdent sont cités. L'ordre dans lequel les contenus sont traités se lit de haut

en bas, de gauche à droite. Pour les rendre plus visibles, nous mettons en italique les objets variables et boucles en italique.

De ce tableau nous dégageons quelques remarques :

- Une variété dans la répartition des objets d'enseignement dans les chapitres dans M_i. Par exemple, l'objet « expressions » est l'objet d'un chapitre entier dans MEF et M3, alors qu'il est seulement dans un sous chapitre chez M2 et il n'est considéré ni comme chapitre, ni comme sous chapitre dans M1.
- Si nous effectuons une répartition d'objets enseignés en Pascal comme suit :
  +) Eléments de base (EB) : alphabet, mots clés, expressions
  +) Règle d'écriture des instructions d'un programme (RE) : déclaration, entrée, sortie, fin, label, blocs
  +) Types de données élémentaires (DE) : nombre entier, nombre réels, constantes, logique, chaîne de caractère, énumération, intervalle
  +) Sous programme (SP) : procédure, fonction
  +) Types de données structurées (DS) : tableaux, ensemble, enregistrement, fichier, chaîne, pointeur

Alors l'ordre dans lequel les notions variable et boucles sont présentées est le suivant :

MEF	EB → RE → DE → SP → *variable, boucle* → DS
M1	EB → RE → DE → SP → tableau, enregistrement → *variable, boucle* → DS
M2	EB → RE → DE → *variable* → DE → *boucle* → SP → DS
M3	EB → RE → DE → *variable* → tableau → *boucle* → chaîne → SP → SP

Si la boucle succède l'introduction des tableaux alors les exemples avec la boucle « for » peuvent être liés avec ce type de données.

- Seul M2 fait référence à la mémoire lors de l'introduction de la notion de variable dans un programme informatique.
- Le programme du MEF introduit l'instruction Goto avant d'autres instructions itératives. Celui du M1 souligne même que les 3 autres instructions itératives peuvent s'écrire en utilisant seulement l'instruction Goto.

MEF

Elément de base
- Symbole de base
- Mots clés
- Noms
- Constantes textuelles

Types de données standards
- Type logique
- Type nombres entiers
- Type nombres réels
- Type caractères

Structure d'un programme
- Déclaration
- Instructions
- Environnement intégré et développé du Pascal 7.0

Expressions
- Constantes
- Opérations
- Appel des fonctions

Instructions
- Instructions simples
  +) *affectation*
  +) *goto*
  +) appel des procédures
  +) entrée et sortie
- Instructions structurées
  +) composée
  +) branchement
  +) *itérative*

Type de données structurées
- Tableau
- Chaîne
- Ensemble
- Enregistrement
- Fichier
- Texte

M1

Symbole et éléments de base
- Alphabet
- Système de chiffres
- Symboles particuliers
- Mots clés
- Noms
- Constantes textuelles
- Fonctions et procédures standards
- Commentaires

Types de données standards et expressions
- Nombre entier
- Nombre réels
- Caractères
- Logiques
- *Variable et expressions*

Structure d'un programme
- Début du programme
- Blocs
- Description, labels
- Détermination des constantes
- Détermination des types
- Description des variables
- Détermination des fonctions et procédures

Types de données simples définies par programmeur
- Type numération
- Type d'intervalle

Types de données structurées
- Tableaux
- Enregistrement

Opérateur (instructions)
- Opérateurs simples
+) *Affectation*
+) *Goto*
+) entrée et sortie
- Opérateurs structurés
+) composé
+) conditionnel
- *Opérateurs itératifs*
+) *For*
+) While
+) Repeat
- Opérateur with

Types ensemble, fichier, pointeur

Organisation d'entrée et sortie, fichiers de textes

Procédure et fonction, paramètre

Exemples sur : pile, file, récursivité, algorithme de tri vertical

M2

Elément de base
- Alphabet, mots clés, noms
- Point virgule
- Commentaire
- Structure d'un programme
- Etapes essentielles pour écrire un programme

Type de données, types de données standards
- Notions élémentaires
- Type logique
- Type nombre entier
- Type nombre réel
- Fonctions standards
- Type caractère
- Type dénombrable et indénombrable
- Extensions sur type nombre entier, nombre réel

Déclaration
- Déclaration constantes
- Déclaration *variables*
- Définition d'un nouveau type de donnée
- Expression
- Instructions : *affectation* etc.
- Instructions composées

Procédure entré et sorite
- Procédure entrée
- Procédure sortie

Instructions conditionnelles
- If…Then…Else
- Case of

Pratique sur Turbo 7.0 et 5.0

Types énumération et intervalle
- Type énumération
- Type intervalle

Boucle déterminé (for) et indéterminé (while, repeat)
- *For*
- *While, repeat*
- *Goto*

Procédure, fonction

Tableaux | **Chaîne**

Ensemble | **Enregistrement**

Fichiers | **Pointeurs**

M3

Introduction
- Histoire des langages
- Langage Pascal
+) Caractéristique
+) Organigramme

Notions, élément de base
- Alphabet
- Mots clés
- Nom, nombre, chaîne de caractère, valeur logique
- Structure d'un programme
- Label, type de données, *variable*,

Types de données
- Types de données
- Type nombre entier
- Type nombre réel
- Type logique
- Type caractère
- Type énumération
- Type intervalle
- Type octet

Expression
- Expression arithmétique
- Expression relation
- Expression logique
- Expression chaîne caractère

Instructions simples
- Instructions
- *Affectation*
- Entrée, sortie

Instruction branchement
- If…Then…Else
- Case…Of

Tableau, instructions itératives
- Tableau
- *Instruction For*
- *Instruction While*
- *Instruction Repeat*

Chaîne | **Sous programme**

Enregistrement | **Fichier**

Ensemble | **Pointeurs**

I.3.2. Définition des notions de variable et de boucle, notation d'affectation

Voici les définitions de la notion de variable et de boucle données dans ces manuels :

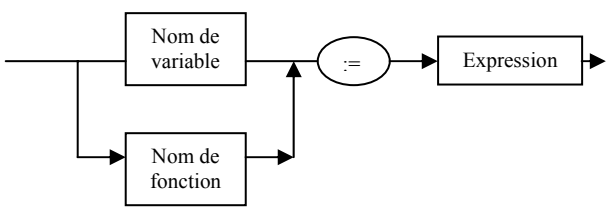
- Variable est un nom, soit déclarée dans la partie de déclaration des variables, soit une partie de la variable tableau, soit une partie de la variable du type structuré (M1, p. 107)
- Variable est une grandeur qui peut changer de valeur. Le nom d'une variable du programme est le nom de la case où est rangée la donnée. (M2, p. 52)
- Les grandeurs participent au processus de calcul dont les valeurs sont changés sont appelées variable, le nom qui correspondent à cette grandeur est le nom de la variable. (M3, p. 137)

On peut noter que ces définitions associent d'abord des variables à des grandeurs, ou le nom de ces grandeurs dont le contenu est susceptible de changer. La relation avec la mémoire est seulement évoquée dans M2.
Regardons comment la notation d'affectation est introduite.
Dans ces 3 manuels cette notation est souvent utilisée sans être explicitée lors de la représentation des algorithmes introduite dans la rubrique « algorithmes » qui se trouve dans la partie « Informatique générale ». Mais dans cette partie, pour noter l'opération des calculs dans un programme tantôt la notation est utilisée, tantôt c'est la notation « = » qui est utilisée à sa place comme le montre l'exemple ci-après :

Pavé de calcul $\boxed{S := S + R}$
Pour illustrer, considérons la résolution de l'équation $ax^2 + bx + c = 0$ […]
Etape 1 : Calculer $D = b^2 – 4ac$ […]
Le schéma correspondant de cet algorithme : […] $\boxed{D = b^2 – 4ac}$ (p. 67)

Notons par là l'incohérence de la notation d'affectation entre ces deux pavés. Il semble que l'affectation est utilisée seulement si la même variable se présente dans deux côtés de l'opération. C'est-à-dire lors qu'il y a le changement de valeur de la même variable dans une opération.
Quelle est alors la définition ? Le tableau de la structuration des cours ci-dessus montre que les deux différents moments : celui de l'introduction de la notion de variable et celui de l'introduction de la notation d'affectation sont complètement séparés.

• Opérateur d'affectation (assigned statement)
Schéma grammatical :
Deux actions sur cet opérateur :
- calculer la valeur de l'expression (à droite du signe « := » ;
- affecter cette valeur à la variable ou à l'appel de la fonction ; (M1, p. 119)



• Affectation est utilisée pour affecter la valeur d'une expression, une constante dans une variable.
L'affectation est notée :=
variable := expression ;
Note : Le membre à gauche est et seulement est une variable. (M2, p. 56)
• Structure grammaticale de l'affectation : V := E
V est la variable ; E est l'expression, dans un cas particulier, E peut être constantes (M3, p. 160)

Dans aucun cas, cette notation est différenciée du signe « = » en mathématique.
L'ordre d'introduire des types de boucles ainsi que la définition d'une boucle sont donnés dans le tableau suivant :

Manuels	Définition	Ordre
M1	Opérateur répétitif provoque l'exécution répétitive d'un groupe d'opérateurs certaine nombre de fois	« Goto » ; « For » ; « While » ; « Repeat »
M2	On rencontre plusieurs fois le cas on doit répéter un travail suivant certaine loi. Le nombre de répétition peut être déterminé d'emblée ou non.	« For » ; « Repeat » et « While » ; « Goto »
M3	Absente	« Goto »; « For » ; « While » ; « Repeat »

On note la présence de la boucle « Goto » dans tous les trois manuels. En suite, c'est souvent dans l'ordre « For », « While » et « Repeat » que ces boucles sont introduites.

En guise de conclusion

- L'algorithmique n'est pas enseignée. Soit les auteurs semblent considérer que les connaissances introductives relevant de l'algorithmique peuvent s'acquérir à travers l'enseignement de la programmation. Soit ils pensent que l'algorithmique n'est pas un objet de connaissance dans l'introduction de l'informatique. Les langages de programmation sont considérés par certains comme langages algorithmiques.
- L'ordre selon lequel les notions de variable et de boucle sont introduites varie d'une université à l'autre.
- La mémoire est rarement présente dans la définition de la notion variable.
- L'équivalence entre des boucles ? Exemples donnés.

II. Etat en France

II.1. Quelques généralités

L'introduction en informatique en première année d'université se fait dans deux diplômes d'études universitaires générales (DEUG) : des mathématiques informatique et application aux sciences (MIAS) ou mathématiques appliquées et sciences sociales (MASS). Le premier s'ouvre à des licences en mathématiques et en informatique et le 2^{e} pour des licences en mathématiques appliquées, en informatique et éventuellement en économie. Contrairement au Vietnam, en France l'élaboration des programmes est à la charge de chaque université. Ces programmes varient d'une école à l'autre. Nous donnons ci-après des remarques générales après la consultation des programmes d'enseignement de quelques universités françaises (Grenoble, Nice, Paris V, VI, VII, Poitiers, Provence, Toulon).

- Cet enseignement se fait sur un volume de temps variant entre 100 et 120 heures sur environ 550 à 600 heures d'enseignement. Il est souvent réparti en deux unités étalées en deux semestres de la première année.
- Il y a plusieurs langages de programmation enseignés à ce niveau : Pascal, C (style impératif) ou Scheme, Calm (style fonctionnel).
- Le système d'exploitation est soit Windows, soit Unix dont l'apprentissage est exigé.
- L'algorithmique est enseigné dans la plupart des universités mentionnées. Cet enseignement est fait avec un langage, soit de programmation, soit algorithmique.

Voici le tableau qui donne les contenus de cet enseignement dans certaines universités :

Université	Intitulé du cours	Semestre	Contenus
Grenoble I Deug MIAS	Informatique	I	Organisation matérielle d'un ordinateur. Rôle du système d'exploitation. Description d'un langage de programmation ; type, affectation ; structures de contrôle, structure de données ; procédure et fonctions. Langage utilisé : Pascal. Environnement : Unix
		II	Types et valeurs, analyse descendante, spécification d'une fonction. Expression conditionnelle. Expression à valeurs booléenne. Evaluation. Constructeurs de type. Définitions récursives. Langage utilisé : Scheme. Environnement Unix
Paris VII Deug MIAS	Langages informatiques	I	- Concept généraux de la programmation (données, ordinateur, mémoire,…) - Initiation à la programmation (tableaux, structures, itération, procédures, notion de récursivité et de pointeur)
		II	- Programmation : récursivité, pointeur - Structure de données : liste, arbre (évaluation et recherche), ensemble
Université de Provence Deug MIAS	Informatique	I	- Généralité sur l'organisation d'un ordinateur : matériel, système d'exploitation, langage. - Système d'exploitation : apprentissage d'un environnement de type Unix. - Algèbre de Boole, codage des entiers en base 2 et fonctions de conversion, circuits arithmétiques et logiques
	Algorithme	II	- Apprentissage de la programmation impérative dans un environnement système de type Unix. - Notion d'algorithme, de langage de programmation, de compilation et d'exécution. Le système d'exploitation et les logiciels. - Données, expression, instructions. - Type et instructions de base : booléens, entiers, réels, chaînes et affectation, conditionnelle, itération. - Types tableaux, enregistrement, ensemble. Fonctions et procédures. Passage par valeur et par adresse. Récursion. - Exemple de résolution de problèmes.
Université de Nice Deug MASS	Informatique pratique	I	- Utilisation d'un clavier, d'une souris, d'un système de multi-fenêtrage. Utilisation d'un éditeur de texte élémentaire (bloc-notes). Outils pour le courrier électronique et la navigation sur Internet. Récupération, archivage et impression de documents. Initiation aux systèmes de traitement de texte et aux tableurs.
	Informatique 2 : Algorithme et programmation impérative	II	- Aspects formel et méthodologiques de la programmation, énoncés et assertions. Structures de contrôle, classes et instances, fonctions et mécanismes de base d'un support d'exécution. Décomposition d'un programme en classe et modules avec interfaces abstraites. Qualité d'un programme, analyse et mesures de performance, utilisation d'une bibliothèque de composants. - Tableaux, consultation de tables, algorithmes de tris. Structurations élémentaires des données, chaînes, articles. - Fichiers séquentiels, fichiers de textes. Arbre et algorithmes récursifs. Techniques d'accès pour des représentations contiguës, chaînées, dispersées ou ordonnées. Algorithmes heuristiques, sur les graphes, de recherche opérationnelle.

Remarques

- L'université Grenoble I annonce le langage utilisé (Pascal et Scheme) mais en réalité, le langage choisi pour cet enseignement est le langage C (l'année scolaire 2004-2005). On peut noter alors qu'ils sont tous des langages du style impératif. Cependant parmi ces langages, Pascal et C sont les plus choisis. Voyons par exemple, les raisons avancées par Granjon :

> Il existe aujourd'hui d'innombrables langages de programmation, plus ou moins répandus, ou plus ou moins dédiés à tel ou tel type d'application. Parmi eux, nous avons retenu, pour guider le lecteur au travers de son initiation, les deux principaux langages traditionnellement choisis dans l'enseignement de l'informatique : le langage Pascal, pour son intérêt pédagogique et sa simplicité de mise en œuvre et le langage C pour son caractère universel et pour la place qu'il occupe aujourd'hui dans le développement de la quasi-totalité des logiciels. (Granjon 1999, p. 2)

- L'enseignement de l'algorithmique est fait dans certaines universités. Des notions comme : analyse et mesure de performance d'un programme, algorithme de tris, algorithmes récursifs etc. sont enseignées. Dans cet enseignement la notion d'organigramme n'est pas présente. Alors avec quel langage utilise-t-on pour écrire des algorithmes ?

II.2. Algorithmique et programmation

Notre examen de quelques livres de cet enseignement montre qu'il y a 2 façons :
+) Enseigner un langage de programmation et les éléments d'algorithmique en même temps ;
+) Enseigner l'algorithmique ;
Nous allons examiner ces deux cas.

Enseignement d'un langage de programmation

Voici le programme d'enseignement en 1ère année de la licence MIAS à l'université Grenoble I, l'année 2004-2005. Ce enseignement, intitulé : Introduction à Unix et à la programmation en langage C comprend 2 parties. La 1ère est une introduction au système d'exploitation Unix et la 2e est consacrée au langage C dont le contenu est décrit comme suit :

1. Cours et exercices : 1.1. Types, variables, constantes, expressions, opérateurs, instructions ; 1.2. Entrées-sorties (printf, scanf, putchar, getchar) : 1.3. Instruction conditionnelle (if, else, else if) ; 1.4. Itération (for, while) ; 1.5. Tableaux à une dimension et chaîne de caractères ; 1.6. Tableaux à deux dimensions ;
2. TPs : 2.1. Instructions conditionnelles ; 2.2. Calcul de la moyenne et de la mention d'un étudiant ; 2.3. Ecriture d'un entier en base 16 ; 2.4. Propagation d'une nouvelle ; 2.5. Tri à bulle ; 2.6. Carrés magiques ; 2.7. Dessin dans la grille ; 2.8. Le jeu du démineur ; 2.9. Labyrinthe ; 3. Projets ; 4. Annexe ;

Comme dans des autres universités, la diffusion du langage C est très liée à celle du système Unix qui est écrit presque entièrement en C. Il est à souligner que le langage C servit de base à plusieurs langages à objets (Objective C et C++) et sa syntaxe de base est reprise dans de très nombreux autres langages, comme Java par exemple. Ceci explique partiellement le choix du langage C et par conséquent, du système Unix dans cet enseignement.
Les notions de variable et de boucle sont introduites dès le début de l'enseignement. On peut noter que à part le type de données élémentaires (nombre entier, nombre réel, caractère etc.) seul le type des tableaux est traité parmi les types des données structurés.
Quelques notions de base en algorithmique comme évaluation d'un algorithme, algorithme de tris sont introduits. Une phase de conception d'algorithme est envisagée dans quelques exemples dans lesquels un langage algorithmique est employé. L'exemple suivant donne une idée de ce langage qui n'est accompagné par aucune définition (p. 53).

Initialiser *continuer* à vrai.
Tantque (*continuer* = = vrai) faire.
  Mettre ici le programme 2
  Demander à l'utilisateur s'il veut continuer.
  Modifier *continuer* si l'utilisateur ne veut pas continuer.
Fin tantque.

A propos du langage Pascal, l'organisation de l'enseignement de ce langage à l'université de Strasbourg montre aussi que les notions de variable et de boucle peuvent être introduites dès le début :

1.Introduction : les logiciels ; organisation de l'ordinateur ; langage de programmation ; 2.Un premier petit programme ; 3.Constante ; 4.Instruction d'affectation ; 5.Les types de variables standard simples et opérateur associés : entiers ; réels ; booléens ; caractères ; 6.Les fonctions standard ; 7.Instruction ; 8.Structure de contrôle : boucle while-do ; boucle repeat-until ; boucle for-do ; instruction If-then-else ; la structure case-of ; 9.Type énumérés non standard ; 10.Les types intervalles ; 11.Tableaux ; 12.Enregistrement ; 13.Procédure et fonction ; 14.Les entrées/Sortie ; 15.Ensembles ; 16.Pointeur ; 17.Corrections des exercices ; (Trau 1997)

Enseignement de l'algorithmique

Nous avons dit plus haut que nous avons repéré deux façons pour cet enseignement :

+) Enseignement l'algorithmique puis un langage de programmation ;
+) Enseignement de l'algorithmique ;
Pour la 1ère, les raisons avancées peuvent être classées en 2 catégories :
- Il faut connaître un langage de programmation afin d'exécuter sur la machine un programme écrit dans un langage algorithmique :

> Pour être compris par une machine, un algorithme doit donc être écrit dans un langage spécifique, qu'on appelle langage de programmation. Un langage de programmation n'étant qu'une façon d'exprimer un algorithme, on comprend aisément que la notion d'algorithme doit être maîtrisée avant d'essayer d'écrire un programme dans un certain langage. (Signac 2004, p. 8)

- Ecrire un programme permet d'avoir l'esprit du développement des logiciels, de comprendre mieux les contraintes de l'implémentation d'un algorithme sur la machine :

> Nous pourrons ainsi aborder la traduction de ces algorithmes dans des langages de programmation, où les contraintes de l'implémentation s'ajoutent harmonieusement à l'analyse algorithmique, sans l'altérer. […] Nous présentons dans cet ouvrage un certain nombre d'exemples de formulations de problèmes, avec leur analyse algorithmique précise et leurs traductions en langage Pascal en langage C. le lecteur pourra ainsi se familiariser avec l'esprit du développement des programmes dans ces deux langages bien différents, qui ont été conçus avec des objectifs distincts. (Cardon et Charras 1996, p.11)

Pour la 2e choix, la conception d'un algorithme est mise en valeur au détriment de la traduction en un langage de programmation :

> En programmant, on se rend vite compte que les plus grandes difficultés de la programmation viennent de la recherche d'algorithmes et non de la nécessité de respecter les particularités syntaxiques (bien qu'elles soient très variables d'un langage informatique à l'autre). (Maysonnave 1996, p. 4)

Dans tous les deux façons, le choix d'un langage algorithmique est justifié. Dans certains livres, un chapitre entier est consacré à définir les règles syntaxiques de ce langage alors dans d'autres, les règles sont implicites.
Le choix d'un langage dans lequel les algorithmes sont écrits est plus ou moins explicité comme par exemple Maysonnave :

> Au lieu d'utiliser un programme pour exprimer les algorithmes que nous souhaitons voir exécuter, nous allons utiliser le langage que nous manipulons avec le plus d'aisance : notre langue maternelle ! Ceci donne la possibilité de s'exprimer sans contrainte de langages (ou presque), chacun avec son vocabulaire personnel. (Maysonnave 1996, p. 4)

Cependant les conventions d'écriture sur ce langage sont données dans son ouvrage au fur et à mesure que des instructions de base comme affectation, itération ou condition etc. sont introduites. Dans d'autre livre, un chapitre entier fait l'objet de ces conventions. L'exemple suivant donne une idée de ce langage (Maysonnave 1996, p. 48) :

Variable x : nombre entier
Début
ECRIRE ''donner l'entier de départ''
LIRE x
Répéter
⌈ *ECRIRE* x
⌊ $x := x * 2$
Jusqu'à x > 100
Fin

Ce langage ressemble au langage Pascal. C'est aussi la manière de procéder la plus utilisée des autres auteurs des livres d'algorithmique.
Voici l'organisation didactique du livre de Maysonnave :

> Chapitre 1. Introduction aux algorithmes : algorithmes ; algorithmes langages et ordinateurs ; écriture des algorithmes ; structures de contrôle ; structure de données ; application aux mathématiques ; utiliser cet ouvrage ; exercices ; Chapitre 2. Variables : généralité ; types de variables ; affectation ; entrées/sorties ; trace d'un algorithme ; constantes ; exercices ; Chapitre 3. Les conditions : introduction ; conventions d'écriture ; exemples ; variables booléennes ; introduction au calcul booléen ; exercices ; Chapitre 4. La

répétition : introduction ; boucle tantque-faire ; boucle répéter-jusqu'à ; boucle pour-faire; exercices ; Chapitre 5. Les tableaux ; Chapitre 6. Procédures et fonctions ; Chapitre 7. Application au calcul numérique ; Chapitre 8. Utilisation des tableaux en mathématiques ; Chapitre 9. Les fichiers séquentiels ;

Les notions de variable et de boucle sont présentées juste après une introduction aux algorithmes. Nous donnons d'abord quelques définitions de la notion de variable puis de boucle.
Concernant la notion de variable :

Données et résultats sont des grandeurs qui sont susceptibles de varier, car un algorithme de traitement a vocation à pouvoir traiter, de manière répétitive, toutes sortes de valeurs différentes. On les appelle, pour cette raison des variables. (Granjon 1999, p. 2)

Une variable sert à stoker une valeur. On peut la représenter par un rectangle à côté duquel on écrit un nom : 33 A BORDEAUX B. La variable de nom « A » a pour valeur le nombre 33, la variable de nom « B » a pour valeur le texte « BORDEAUX ». (Maysonnave 1996, p. 8)

Une variable est une case mémoire (qui a une certain taille) et qui peut éventuellement prendre une valeur par affectation (x ← 2) (Cours « Introduction à l'informatique », UJF, Grenoble 2004)

On note que ces définitions font, pour la plupart la relation avec la mémoire « une variable est une case de mémoire ».
La notation est introduite explicitement dans tous les manuels étudiés et en même temps de l'introduction de la notion de variable. Les auteurs font attention à distinguer cette notation avec le signe « = » en mathématique.

- D'une manière générale, il s'agit d'évaluer une expression comportant des variables, des constantes, des opérateurs (+, -, /, *, etc.) et des fonction plus sophistiquées (qui font partie des langages ou que nous apprendrons ultérieurement à construire). Les résultats des évaluations de ces expressions sont la plupart du temps « rangé » dans des variables. On dit qu'il s'agit d'une affectation du résultat d'une expression à une variable. […] En algorithmique, nous ferons systématiquement la différence entre le concept d'égalité et celui d'affectation qui traduit bien le « devient égal à ». On préféra écrire « := », pour cette affectation, comme c'est le cas dans certains langages de programmation. (Granjon 1999, p. 3)

- C'est l'opération fondamentale sur les variables, elle consiste à changer la valeur de la variable. Nous la noterons par le symbole « := » et nous écrirons :
- à gauche de ce signe, le nom de la variable,
- à droite, une expression fournissant sa nouvelle valeur.
Le symbole « := » sera lu « reçoit » ou « prend pour valeur », mais pas « égal » pour être sûr de ne pas le confondre avec l'égalité des mathématiques qui est autre chose ». (Maysonnave 1996, p. 10)
- L'instruction d'affectation, notée « ← », permet de stocker une valeur dans un emplacement mémoire. L'instruction x ← 123 stocke la valeur 123 dans l'emplacement mémoire x. On dit qu'on a affecté la valeur 123 à la variable x. (Cours « Introduction à l'informatique », UJF, Grenoble 2004)

L'ordre d'introduction des types de boucles ainsi que la définition d'une boucle sont donnés dans le tableau suivant :

Manuels	Descriptions	Ordre
Granjon p. 26	Une itération consiste à exécuter un bloc d'instruction un certain nombre de fois. On parle d'une boucle.	« For » ; « While » ; « Repeat »
Maysonnave p. 44	Nous allons examiner différentes manières de faire répéter une séquence d'instructions, appelée corps de la boucle.	« While » ; « Repeat » ; « For »
UJF Grenoble	Les instructions d'itération permettent de répéter une action (ou une séquence d'actions).	« For » ; « While » ;

On note d'abord l'absence de la boucle « Goto ». La plupart des manuels introduisent les trois types de boucles suivant cet ordre : « For », « While » et « Repeat ». Il semble que la boucle « For » est introduite d'abord par l'absence d'une explicitation sur la condition pour sortir la boucle. Autrement dit, on connaît le nombre d'itérations à l'avance ce qui n'est pas le cas pour deux autres boucles.

III. Conclusion comparative

- Au Vietnam dans un enseignement de langage de programmation, l'apparition des notions de variable (dans une affectation) et de boucle (dans des instructions itératives) vient assez tard. Le moment de cette introduction varie d'un livre à l'autre. Alors qu'en France, cette introduction est faite juste après l'introduction du langage de programmation.
- En France, des notions d'algorithmique sont présentées dans tout enseignement, que soit dans celui d'un langage de programmation ou celui d'algorithmique. Un langage algorithmique est toujours introduit. Dans cet enseignement, on peut remarquer aussi que les notions de boucles et de variables sont introduites presque depuis le début. Au Vietnam, il n'y pas l'enseignement d'algorithmique dans l'introduction de l'informatique. Certains auteurs assimilent des langages de programmation aux langages algorithmiques.
- Au Vietnam, le seul style de programmation est le style impératif et le langage Pascal est le seul langage de programmation enseigné. En France le style de programmation dominant est aussi le style impératif mais plusieurs langages sont enseignés comme Pascal, C, Java etc. Les organigrammes sont présents dans plusieurs livres alors qu'ils sont rarement présents dans les livres utilisés en France.
- En France, avec les types de données élémentaires (nombres entiers, nombres réels, chaînes de caractère etc.) un autre type de données, les tableaux, est abordé dans tout l'enseignement de programmation et d'algorithmique. D'autres types de données comme : enregistrement, fichiers séquentiels etc. sont abordés mais leur présence varie d'un livre à l'autre. Alors qu'au Vietnam ces types de données et d'autres comme : ensemble, pointeur etc. sont tous présents dans l'enseignement.
- En ce qui concerne les boucles :

+) Dans l'enseignement d'algorithmique les deux types de boucle : tantque-faire et compteur, sont enseignés. Plusieurs livres d'algorithmiques présentent le 3^{e} type de boucle : répéter-jusqu'à. Aucun ne présente la boucle « aller à ».

+) Dans l'enseignement de programmation, tous les trois types de boucle sont abordés sauf la boucle aller à. Au Vietnam, ce type de boucle est présenté avant les 3 autres types de boucles dans quelques livres (cf. M2).

- Le langage est soit un langage de programmation évolué, soit un langage algorithmique qui ressemble à un langage évolué. Ce langage, comme nous l'avons vu, ressemble à un langage structuré du style impératif. Le langage enseigné est ainsi représentatif d'une classe de langages existants. Ceci exige des connaissances préalables d'un langage de programmation à l'enseignement de l'algorithmique
- La machine n'accompagne pas cet enseignement dans la mesure où les notions enseignées comme variable, boucle ne sont pas introduites en présence de la machine, autrement dit on peut attester l'absence de machine de référence.

Or l'analyse de la genèse d'un enseignement d'informatique montre une autre stratégie dans laquelle une machine de référence, réelle ou fictive, est présente. L'enseignement de l'algorithmique est alors lié à la compréhension de l'architecture de la machine et à la conception d'un langage approprié à cette machine. A l'heure actuelle, Knuth plaide pour cette stratégie :

> But the reasons for machine language that I gave in the preface to Volume 1, written in the early 1960s, remain valid today:
> - One of the principal goals of my books is to show how high-level constructions are actually implemented in machines, not simply to show how they are applied. […]
> - The programs needed in my books are generally so short that their main points can be grasped easily.
> - People who are more than casually interested in computers should have at least some idea of what the underlying hardware is like. Otherwise the programs they write will be pretty weird.
> - Machine language is necessary in any case, as output of many of the software programs I describe.
> - Expressing basic methods like algorithms for sorting and searching in machine language makes it possible to carry out meaningful studies of the effects of cache and RAM size and other hardware characteristics (memory speed, pipelining, multiple issue, look aside buffers, the size of cache blocks, etc.) when comparing different schemes.

> Moreover, if I did use a high-level language, what language should it be? In the 1960s I would probably have chosen Algol W; in the 1970s, I would then have had to rewrite my books using Pascal; in the 1980s, I would surely have changed everything to C; in the 1990s, I would have had to switch to C++ and then probably to Java. In the 2000s, yet another language will no doubt be de rigueur. I cannot afford the time to rewrite my books as languages go in and out of fashion; languages aren't the point of my books, the point is rather what you can do in your favourite language. My books focus on timeless truths. (Knuth 2005)

La présence d'une machine et la compréhension de son architecture, suivant le principe de Von Neumann, dans une introduction de l'informatique semble jouer un rôle crucial dans la genèse des objets de base en informatique ainsi que la genèse d'un langage de programmation (style impératif) lors de cet enseignement.

- On peut noter cependant que les objets variable et boucle constituent des objets de base dans un enseignement de programmation du style impératif. Ils sont aussi transversaux tout le long de tel enseignement. Une étude épistémologique de ces objets s'avère nécessaire dans l'introduction de l'informatique dans EMS.

Annexe des chapitres B2 et B3

I. Exemple d'un programme à cartes pour la machine Analytique

Nous avons vu dans le chapitre B2 que la notion de programmation informatique n'est apparue qu'à partir de la machine Analytique de Babbage avec la communication entre l'homme et la machine par des programmes à cartes. Cette technique est inspirée des métiers à tisser de Jacquard. Une telle communication « permettait, au commencement d'un travail et à la fin de chaque phase de ce travail, d'indiquer la séquence d'opérations constituant la phase suivante » (Moreau 1987, p. 15). La première communication (par un programme) entre homme et machine est née à l'aide d'un dispositif de cartes perforées dont chacune a son propre usage (cf. Walker 2003). Nous avons vu aussi un « programme » avec boucle écrit par Ada à telle machine. Nous allons voir un programme avec des instructions cartes pour cette machine.
Rappelons que la machine Analytique comprend : dispositifs d'entrée (lecteur de cartes) de sortie (impressions) et 3 unités : mémoire, unité de commande (ou de contrôle) contenant et unité de calcul qui contient, entre autres, des registres (qui sont : ER « Ingress Axes », SR « Egress Axes », et PR « Primed Axes ») et un dispositif, appelé « run-up » levier pour que la machine s'arrête lorsque certaines conditions sont satisfaites durant le processus de calcul.
Concernant les cartes, il est utile de rappeler les 3 types de cartes dont Babbage se sert pour les programmes cartes :
- Cartes d'opération (+, -, × ou *, ÷ ou /) : elles consistent à commander le moulin à réaliser des opérations variées d'arithmétique à savoir addition, soustraction, multiplication et division ;
- Cartes de nombres (N) : elles fournissent des constantes numériques quand la mémoire l'exige. Des nombres de ces cartes sont souvent le résultat des calculs précédents ; chaque carte contient la lettre N suivie d'un numéro de la colonne dans la mémoire dans laquelle ce nombre est placé ;
- Cartes de variables (L, Z, S) : elles transfèrent des valeurs depuis la mémoire (L et Z) dans le moulin pour se servir d'un argument dans des opérations, et transfèrent le résultat d'un calcul par le moulin dans la mémoire (S) suivant les règles suivantes :
  +) Carte L : Transférer un nombre depuis la mémoire à ER dans l'unité de calcul, et garder une copie de ce nombre dans la mémoire ;
  +) Carte Z : Transférer un nombre depuis la mémoire à ER dans l'unité de calcul, et mettre à zéro la mémoire ;
  +) Carte S : Transférer un nombre depuis SR dans l'unité de calcul à la mémoire (sur une colonne;

Si le symbole « ' » précède de ces cartes il s'agit alors du transfert à/depuis PR concernant.

Pourquoi Babbage doit-il distinguer les cartes nombres et les cartes variables ? Nous allons essayer de trouver la raison dans un programme avec boucle.
Le programme suivant qui calcule la factorielle de six utilisant une boucle permettrait d'avoir une vue plus près de ces cartes (cf. Walker 2003) :

N°	Instructions cartes	Actions internes de la machine et produites par chaque instruction	Codage moderne
1	N0 6	Placer le nombre 6 dans la colonne 0 la mémoire	Lire 6
2	N1 1	Placer le nombre 1 dans la colonne 1 la mémoire	Lire 1
3	N2 1	Placer le nombre 1 dans la colonne 2 la mémoire	Lire 1
4	×	Activer la multiplication du moulin	
5	L1		6 → L1
6	L0	Multiplier N1 par N0	1 → L0
7	S1	Placer le produit dans la colonne 1 la méoire	L1 × L0 → S1
8	-	Activer la soustraction du moulin	
9	L0		
10	L2	Soustraire N0 de N2	
11	S0	Placer le résultat dans la colonne 0 la mémoire	L0 – L2 → S0

12	L2		
13	L0	Soustraire N2 de N0	L2 – L0
14	CB ? 11	Si le résultat est encore négatif alors retourne 11 lignes à partir de la ligne présente, donc revenir à la ligne 4.	Si (L2 – L0) × L2 < 0 alors retourner à 4 sinon arrêter

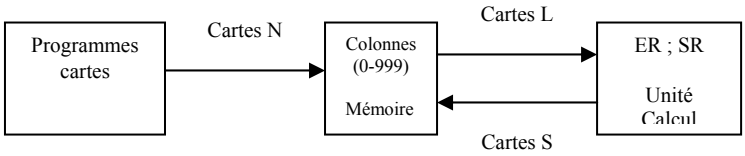
Le fonctionnement du levier « run-up » lors de la soustraction et de la carte éclaire davantage ce programme :

> Subtraction. The value in the second Ingress Axis is subtracted from that in the first (ignoring the contents of the Primed Ingress Axis), and the difference is place on the Egress Axis. If the result of the subtraction differs in sign from that of the first argument, or a borrow-in occurs (resulting from an overflow of the 50 digit capacity of the Mill), the run-up lever is set.
> B : Back (skip backward and repeat cards in the reader).
> ? : Advance (“F”) or back (“B”) only if the Mill’s run-up lever is set. (Walker 2004)

Ces cartes en s’associant avec le levier de l’unité de calcul peuvent être interprétées comme la boucle du type « Si…alors retourner à… ».
Etant donné que seule la carte combinatoire CB permet de répéter un ensemble de cartes, on peut déduire que la machine Analytique permet seulement ce type de boucle. En dehors de la soustraction, les conditions de deux autres opérations arithmétiques pour lesquelles levier « run-up » est actionné sont :
+) Addition : Si le signe de la somme est différent du celui du 1er terme ;
+) Division : Si le quotient a plus de 50 chiffres ou le diviseur est égal à zéro ;
Regardons maintenant le changement de valeurs des variables L0, L1, L2, S0 et S1du programme. L’exécution de chaque boucle donne les valeurs à ces variables comme suit :

N°	L0	L1	L2	S0	S1	L2 – L0
1	6	1	1	5	6	1– 6 = - 5 < 0 (Continuer)
2	5	6	1	4	30	1 – 5 = - 4 < 0 (Continuer)
3	4	30	1	3	120	1 – 4 = - 3 < 0 (Continuer)
4	3	120	1	2	360	1 – 3 = - 2 < 0 (Continuer)
5	2	360	1	1	720	1 – 2 = - 1 < 0 (Continuer)
6	1	720	1	0		1 – 1 = 0 (Finir la boucle)

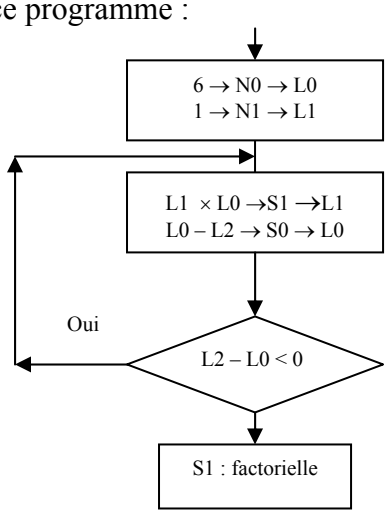
Le transfert des nombres par différentes cartes depuis l’entrée (le programme carte) et deux autres unités (mémoire et calcul) durant l’exécution d’un calcul est indiqué dans le schéma suivant :



Transfert des nombres durant un calcul

Par exemple, le nombre 1 : ce nombre est d’abord entré dans la mémoire grâce à la carte N1, il est placé dans la colonne 1 de la mémoire. Puis pour préparer le produit 1 × 6 (les 3 instructions de cartes : « × L1 L0 »), ce nombre est transféré dans le registre d’entrée ER de l’unité de mémoire par la carte L1 qui laisse cependant cette valeur intacte sur la colonne 1. Le résultat, le nombre 6, est ensuite place dans le registre de sortie SR. Puis l’instruction carte S1 le transfère depuis SR à la colonne 1 dans la mémoire. Cette colonne 1 change donc la valeur de 1 à 6. Au cours du calcul, elle reçoit successivement les valeurs 1, 6, 30, 120, 360 et 720 et passe ces valeurs à l’unité de calcul au moyen des cartes S et L. Elle joue alors le statut de la variable informatique. Or c’est la carte N1 qui enregistre ce nombre dans la mémoire. Elle est alors une variable dans un programme informatique. Les valeurs contenues dans la carte L2 lors de transferts ne changent pas. Elles sont égales toujours à 1. Alors les nombres les termes « carte nombre » et « carte variable », utilisés par Babbage n’ont pas le même sens que le terme « variable » dans un programme informatique d’aujourd’hui.

Voici l'organigramme de ce programme :



Le tableau précédent montre que les valeurs contenues dans les cartes L0, S0, L2, S2 changent. Dans un codage plus moderne d'un langage évolué, avec le même type de boucle ce programme s'écrit ainsi :

1. 6 → C {Initialiser la variable compteur}
2. 1 → S {Initialiser la variable factorielle}
3. C × S → S {Calculer la factorielle}
4. C – 1 → C {Décrémenter la variable compteur}
5. Si C > 0 alors retourner à 3
6. Imprimer S {Sortir le résultat}

Ce programme exige 2 variables (C et S) et une constante (le nombre 1).
Nous avons vu dans la partie précédente que le jeu d'indices d'états utilisé dans l'écriture d'un programme à une machine à mémoire mais fictive peut permettre l'émergence d'une notion de boucle mais ne favorise pas encore l'émergence de la notion de variable en informatique. L'exemple qui vient d'être examiné montre aussi qu'avec une machine réelle cette écriture en un langage compris directement par cette machine ne permet pas non plus cette émergence. Est-ce que le programme, dans lequel les transferts des données entre différentes unités dans par cartes sont indiqués, compris directement par la machine peut diluer la notion de variable ?
L'analyse de l'émergence de la notion variable liée à trois aspects suivants pourra fait l'objet d'une étude dans l'avenir :

- L'aspect dynamique vs l'aspect statique ;
- L'aspect matériel vs l'aspect abstrait ;
- L'aspect langagier évolué vs l'aspect langagier de base niveau ;

II. Notes sur les langages de programmation

II.1. Panorama de différents langages de programmation

Depuis la naissance du premier langage de haut niveau, Fortran, en 1954, des milliers de langages de programmation ont vu le jour. A titre de comparaison en 1967, il n'y avait que 120 langages (dont seulement 15 étaient vraiment utilisées) et en 2003 plus de 2500 langages sont référenciés (cf. Hunault 2004). Il y a plusieurs manières pour classifier ces nombreux langages dont quelques exemples sont donnés ci-dessous :
- Dans son ouvrage Tucker classe onze langages qu'il juge principaux, selon 3 groupes : langages traditionnels (Pascal, Fortran, Cobol, LP/1), langages plus spécialisés et de style divers : Snobol, Apl et Lips, langages de conception et de tendances plus récents : Prolog, C, Ada et Modula-2. (cf. Tucker 1986) ;
- Hunault (2005) suggère quatre paradigmes pour classer les langages de programmations :

+) Objectif du langage : calcul scientifique (Fortran, Apl), calcul mathématique formel (Maple, Derive), gestion (Cobol, Foxpro), bases de données (Dbase, Sql), logique (Proloque), etc. ;
+) Type de traitement : via la compilation comme C, Pascal, Ada, etc. ou l'interprétation comme : Javascrip, Tcl/Tk, Apl, etc. ;
+) Importance du langage : langages de script (Awk, Rexx, Perl, Tcl), langages d'interrogation (Sql, Dbase, Sasl), langages d'interfaces graphique (Tk) ou d'interface Web (Javascript, Php), langages quasi-universels ou ciblés (C, C^{++}, Pascal, Ada, Lisp, etc.) ;
+) Dans un langage de programmation, on peut soit indiquer comment faire avec un langage impératif (langages peu structurés comme Basic, Fortran ; langage structurés par blocs comme Pascal, C ; langages conçus comme langage objets Small Talk, C^{++}, Java, Ada), soit indiquer ce qu'on veut faire avec un langage déclaratif (langage fonctionnels comme Lips, logiques comme Prolog) ;

II.2. Brève histoire des langages de programmation du style impératif

Backus, avec son équipe de recherche au sein d'IBM, écrit le compilateur pour le premier langage évolué Fortran (Formula translation) en 1954 :

> C'est le premier langage informatique de haut niveau, c'est à dire qu'il nécessite un programme intermédiaire (le compilateur) qui le traduit en instructions compréhensibles par l'ordinateur. L'avantage est que le programme en FORTRAN est indépendant de la machine, il suffit d'avoir le compilateur adapté. Il est encore utilisé dans les domaines scientifiques et techniques. (Guillier 2005)

Avec l'apparition du premier langage évolué, la façon d'écrire des programmes change énormément :

> En 1957 […] plus personne ne mettait en doute l'efficacité des compilateurs, la communauté scientifique, passant alors d'un extrême à l'autre, pris l'habitude de définir in abstracto les langages de programmation, faisant hypothèse implicite que, s'il en était besoin, il serait toujours possible d'en rédiger un compilateur. (Moreau 1987, p. 170)

En 1958 un groupe de chercheurs s'est crée en Europe pour définir un langage « totalement indépendant d'une réalisation éventuelle sur une machine » (Moreau 1987, p. 1974). Après un une rencontre avec d'autres chercheurs américains ce groupe a définit le langage Algol en 1958. En 1960, apparu la publication du cahier des charges du langage de programmation Cobol. Il devient, après le Fortran, le second grand langage de programmation universel, faisant ainsi rapidement disparaître l'Algol. En 1964 Kurtz et Kemeny créent le langage Basic pour leurs étudiants. L'année 1968 marque la création du langage Pascal par Wirth qui est utilisé dans l'enseignement de programmation. Le langage C, crée en 1973 par Ritchie et Thompsonest, à la fois proche de la machine permettant de manipuler directement les adresses de la mémoire et suffisamment généraliste, le rendant ainsi facilement portable. En 1979, le langage Ada, développé par Ichbiah de la société Française Bull va être choisi par le l'armée américaine comme l'unique langage de développement imposé à ses services. Les années 80 marquent la naissance du concept de programmation orientée objet avec Smalltalk-80 de Kay et Alto de la compagnie Xerox (Etats-Unis) qui donne la version C^{++} et Objective C. Dans les décennies 1980 et 1990, de nouveaux langages impératifs interprétés ou semi-interprétés doivent leur succès au développement de scripts (série d'instructions servant à accomplir une tâche particulière) pur des pages web dynamiques et applications client-serveur. On peut citer dans les catégories Perl (Wall 1987), Python (Van Rossum 1990), Php et Java (Sun Microsystèms 1996). (cf. Rossi 2005, Wikipédia 2004)
Quelle est l'évolution des écritures de boucles, de variable à travers ces langages ?

II.3. Le langage Fortran

Nous nous attardons sur le langage Fortran, crée par Backus et son équipe en 1954, car c'est le premier véritable langage évolué et le premier langage indépendant de la machine (cf. Moreau 1987). Ce langage, via la version Fortran 90, est toujours utilisé dans le milieu scientifique à l'heure actuelle. C'est aussi un langage qui « plonge ses racines dans l'histoire des langages de

programmation » (Tucker 1986, p. 9). Nous nous intéressons aux notions de boucle et de variable dans l'évolution ce langage.
Depuis sont apparition en tant que premier « langage de haut niveau » sur l'IBM 704 (l'un de concepteur de cette machine est aussi Backus) ce langage a beaucoup évolué et connu plusieurs versions comme Fortran I (en 1954), Fortran II (en 1957), Fortran III (1958) et Fortran IV (en 1962), avec des modifications (notamment en 1977 et en 1990), qui donne le nom Fortran 77 et Fortran 90 et qui est encore en usage d'aujourd'hui.
Dans ces versions, les structures de boucles dans ce langage ont évolué sans cesse.
Pour les versions de Fortran I et II, deux boucles sont présentes :
- Boucle « If *condition* Goto n1 *instructions* Goto n2 »
- Boucle « Do n1 I = n2, n3 »
Voici deux exemples de deux programmes initialisant les 5 valeurs d'un tableau A(I) à 5, 4, 3, 2 et 1 (cf. Tuckey 1986).

Boucle « Goto » :

```
I = 1
10  IF (I.GT.5) GO TO 20
A(I) = 6 – I
I = I + 1
GO TO 10
20  _____
```

Boucle « Do » :

```
      DO 10 I = 1,5
10      A(I) = 6 - I
```

Par rapport à des langages machines à l'époque, comme par exemple, le langage assembleur pour la machine EDVAS (cf. chapitre 2.A), ce premier langage évolué présente une amélioration très nette.
Notons que la notation d'affectation est écrite avec le signe égale : I = I + 1. La comparaison « If a = b » est exprimé en ce langage par « If (a .EQ. b) ». Cette manière de noter l'affectation et la comparaison ne change pas à travers les différentes versions du langage.
La première boucle est du type « aller à » qui porte la trace de la programmation non structurée à l'époque. La 2^{e} est la boucle « compteur » mais l'usage nécessite encore la numérotation de ligne pour marquer l'endroit où les instructions doivent être exécutées. Les boucles « tant que » et « répéter » ne peuvent être écrites que sous la forme la boucle « aller à » avec la numérotation de ligne. Dans les versions Fortran III et IV, cette forme de la boucle « do » est améliorée. Il est possible désormais d'exécuter un ensemble d'instructions à l'intérieur d'une boucle ainsi que des boucles emboîtées. La version Fortran 77 s'ajoute des fonctions améliorant son usage en programmation structurée (cf. Tucker 1986). Cette version propose alors le block de l'instruction, par exemple « If …Then…else…endif ». Avant Fortran n'offrait que la forme « If…Goto » instruction. L'écriture de la boucle « do » est présentée dans Fortran 77 comme suivante :

```
DO n1  I = n2, n2, Pas
   instructions
n CONTINUE
```

Alors l'ensemble d'instructions du corps de la boucle est limité entre Do et n Continue. Sur le langage existant Fortran à l'époque, Tucker remarque que « l'absence d'un ensemble complet de structures de contrôle (essentiellement le précieux While), et la présence d'une syntaxe à la ligne » handicape un peu la programmation en ce langage (Tucker 1986, p.268). Les versions plus récentes comme Fortran 90 ou Fortran 95 possèdent plusieurs améliorations dont les délimiteurs dans les blocs d'instructions « Do…Enddo », l'abandon de numérotation à la ligne, l'instruction « Exit » à l'intérieur d'une boucle permettant alors la formulation de deux boucles «tant que » et « répéter ». Et ce, pour tout compilateur de ces versions. Dans des versions plus récentes, suivant des compilateurs on peut y avoir des boucles « while *condition* do instructions enddo », « do instructions while *condition* » ou « do instruction until *condition* ». La boucle

« while » est acceptée seulement dans des compilateurs de Fortran 77 incluant WATFIV-S. (cf. Tucker 1986)
L'exemple suivant fait la somme des 5 premiers nombres pairs :

Boucle compteur « Do » :

```
SOMME = 0
DO I = 10, 0, 2
SOMME = SOMME + I
ENDDO
```

Boucle tant que « Do » :

```
SOMME = 0
DO WHILE I .LE.10
   SOMME = SOMME + I
   I = I + 2
ENDDO
```

La boucle « répéter » n'est pas présente dans Fortran 90. On peut cependant l'écrire à l'aide de la boucle Do avec l'instruction Exit : « DO Instructions IF… EXIT Instructions ENDDO ».
- La prédominance de la boucle compteur dans ce langage ;
- Une amélioration : boucle « go to » → boucle do (avec numérotation de la ligne) → l'abandon de la numérotation de la ligne, l'ajout de la boucle « do…while » et l'ajout des délimiteurs pour un bloc d'instruction.

Annexe du chapitre D1

I. Des protocoles d'enseignants

I.1. Protocole de la situation 1 avec groupe 2 de la classe 2nd F1

Légende () : notes de transcripteur ; // : inaudible ou la transcription n'est pas sûre; (…) silence long ;… silence court ; Ob. observateur ; Ppu : P s'adressant à la classe ; Ppr : P s'adressant à un(e) élève ou un binôme ; E : un(e) élève. Es : plusieurs élèves en même temps ;

1. Ppu : Alors, voilà, il y a deux postes particuliers donc que je voudrais que ce soit Ingrid et Karen qui viennent là, au hasard.
2. Es : Oh, non. Oui, au hasard.
3. Ppu : Et alors là, vous allez être trois, Sandy, Elsa et Delphine. Vous vous mettez au 3 postes. Il y en a une qui va mettre là, deux là et ensuite vous allez travailler ensemble. Voilà. Puis les autres, vous rentrez dans la salle et puis vous vous répartissez sur les postes. Je vais vous expliquer ce qu'on va faire. Bonjour. Voilà, merci. Bonjour. Voilà, vous vous mettez sur les postes. Vous ne touchez pas pour l'instant aux touches et au clavier. Vas-y, vas-y, passe. Bonjour. C'est un laser, c'est très… Bonjour…Alors, d'abord, vous touchez pas aux écrans pour l'instant. Vous allez juste besoin d'un stylo, tout est fourni, les feuilles sont fournies, c'est, c'est…parfait. Donc je vous présente donc M. Birbent, M. Nguyen et Mme. Bessot qui donc font la recherche sur l'enseignement des mathématiques. Et donc, qui nous proposent euh, enfin qui font un travail sur le, l'utilisation de la calculatrice. Donc je vais vous donner des énoncés. Sur ces énoncés, il y a deux exercices. Le 1er exercice comporte 2 questions à faire seul, ah ? Chacun pour soi pratiquement ah ? Vous avez droit d'échanger de petites choses mais chacun fait son exercice sur sa calculatrice qui se trouve sur l'ordinateur, je vous expliquerai après comment on les démarre. Et puis dans un 2e temps, vous vous mettez par binômes, ou par trinômes, les trois-là. D'accord ? Et vous aurez un exercice à faire en binôme. Je vous re-expliquerai en détail comment ça fonctionne. Alors, la 1ère chose à faire, c'est de démarrer la calculatrice. Donc, démarrer la calculatrice, vous faites comme moi. Sur le, sur le bureau-là, vous devrez avoir un raccourcie calculatrice point bat, normalement. Tout le monde a ça ? Voilà. Vous cliquez 2 fois dessus. Il y a un écran noir qui s'ouvre. Et puis une fenêtre où vous allez rentrer votre nom, prénom, la classe, l'établissement et l'année scolaire qui est 2003…Pardon. Tiens Maxime.
4. E : On met LMG ?
5. Ppu : Voilà LMG, vous pouvez le mettre ah. Mettez le groupe pour la classe. Bon, enfin, il y a votre nom déjà. C'est bien. Année scolaire, voilà. Vous pouvez taper, vous pouvez valider, une fois que vous avez fait ça. J'ai pas mis 2 là. Ca c'est.
6. Ppr : Ca c'est Alexandre, c'est du groupe précédent. Il me manque un énoncé, mais c'est peut-être…
7. Ppu : J'en ai pas mis peut-être deux à quelqu'un ? Il y a quelqu'un qui en a deux ? Un chacun. Non, non, il en faut un chacun. Oui, oui, je veux bien mais. On en prend un là. Tiens. Voilà. Je fais la même chose que vous, donc.
8. E : Il y en a deux !
9. Ppu : Et, bah, voilà. Donc je vais faire comme vous (il vient à l'ordinateur public). Second D. Groupe B, classe 2e D, établissement LMG. 2003. Ppu : Voilà, il y a une chose importante. Pour pouvoir utiliser la calculatrice…On attend que Caroline ait démarré… Vas-y.Vas-y. Je vais y passer après. Donc, pour pourvoir utiliser la calculatrice, vous allez en haut dans « séance », en haut à gauche, dans le menu « Séance ». Vous tapez, bah, vous cliquez sur le nouvel exercice et vous rentrez le numéro de l'exercice 1.
10. E: On met 1 ?
11. Ppu : Exercice 1, le numéro 1 ah ? Le chiffre 1. D'accord ? Ensuite, il faut retourner dans « séance »…et dire première question. D'accord ? Et quand vous passerez à la 2e question, avant de passer à la 2e question, vous y penserez bien : il faudrai y revenir et dire je passe à la 2e question.
12. E : Là, on dit oui ?
13. Ppu : Là vous dites oui.
14. E : Et on peut commencer ?
15. Ppr : Et vous pouvez commencer l'exercice et la 1ère question qui vous a proposée.
16. Ob : Il faut appuyer sur AC.
17. Ppr : Pardon ? Ah, oui, pour démarrer la calculatrice, il faut tapez sur AC/On. Vous voyez la touche ? C'est comme sur une calculatrice normale ah ? Ca fonctionne pareil.
18. Es : Il y a pas des parenthèses. Il y a pas des parenthèses ?
19. Ppu : Alors, il y a certaines touches qui marchent et certaines touches qui ne marchent pas. Donc, qu'il faut vous vous débrouilliez. C'est peut-être que ce soit fait exprès. C'est en fait ça.
20. Ppr : Tu vois le brouillage numérique. Tu valides. Voilà, ça va démarrer. Voilà séance, vay-y, je te laisse faire. Séance, nouvel exercice. Tu

rentres séance, le numéro d'exercices, tu mets exercice 1. Séance, de nouveau, première question, voilà et tu mets nouvelle question. Voilà, d'accord ? (à un autre groupe) Vous mettez bien vos noms sur les énoncés ah ?
21. E: Monsieur, comment fait que je peux pas effacer.
22. Ppr : Comment ? Tu veux effacer ? Alors, c'est quoi la touche pour effacer ? Voilà, il y ça, soit t'effaces tout ou avec ceci, t'effaces un caractère par caractère, d'accord ?
23. E : On peut arrondir ou pas ?
24. Ppr : Vous faites ce qui est demandé dans l'exercice, je, je fais pas de commentaire.
25. E: Comment on peut revenir en arrière ? Pour un chiffre ?
26. Ppr : Alors, comment on fait sur une calculatrice normale ?
27. E : Bah, avec C. Mais ici c'est C/Off.
28. Ppr : Ouais. Essaie, faire…En fait tu vois, il y a deux fonctions là. Tu vois, il y a deux fonctions là. Tu vois avec la 1ère et avec second, c'est avec la touche seconde là, d'accord ? Essaie. Vas-y.
29. E : Ah, oui (…)
30. Ppu : Quand vous passez à la 2e question, vous n'oubliez pas de l'indiquer à la calculatrice en lui disant « séance », « 2e question » ah ?
31. E : Mais là, j'ai pas à faire nouvel exercice là.
32. E : J'ai la question suivante.
33. E : Voilà.
34. Ppr : Mais, non, tu as quelque chose pour effacer ah ? Si tu veux effacer l'écran.
35. E : A mais non, je veux pas effacer mais ça fait rien.
36. E : A chaque fois, il faut refaire tout ça ?
37. Ppr : Il faut faire ce calcul.
38. E : Ah, d'accord.
39. Ppu : Et alors, vous voyez dans la question 2, il y a une chose importante, c'est qu'on va vous demander de compter le nombre de touches que vous aurez utilisé. Le nombre de fois que vous aurez cliqué sur une touche de la calculatrice.
40. E : Dans le 1er exercice, j'ai fait « nouvel exercice » t'en as fait deux alors…c'est quoi ? Alors qu'il faut 2e question.
41. Ppr : Alors, on va faire 1ère question oui, et on va faire nouvel exercice. Oui. Tu fais un bis. 1, bis. Ca veut dire que tu refais l'exercice 1. Comme ça, on s'y retrouvera, la question suivant. Tu veux passer à la question 2, c'est ça ?
42. E : Oui, c'est ça.
43. Ppr : Voilà, question 2, c'est ça.
44. E : Monsieur, comment on faire pour mettre au carré ?
45. Ppr : Est-ce qu'il y a une touche carrée ?
46. E : Non, ah, oui mais…
47. Ppr : Elle marche pas ! Donc il faut te débrouiller.
48. E : Monsieur, comment revenir ?
49. Ppr : Vous êtes passé…Parce que t'est déjà passé à la 2e question ? Non, c'est pas grave. Essaie de …
50. E : Il faut faire nouvel exercice ?
51. Ppr : Non, non. Il faut faire question suivante. Vous êtes toujours dans le même exercice.
52. E : Ah, oui, alors que je fais nouvel exercice.
53. Ppr : Alors, on, on va faire. T'as mis nouvel exercice déjà ?
54. E : Oui.
55. Ppr : Alors, ce qu'on va faire, on va recommencer l'exercice. 1 bis, pour indiquer qu'on revient sur l'exercice 1 en fait, d'accord ? Et puis, on va lui dire question…suivant. Voilà
56. E : OK. Et là, en fait on va compter le nombre de fois qu'on a fait et tout. Par exemple, j'ai tapé 3, 6, 9…
57. Ppr : Oui, voilà, y compris les chiffres ah, d'accord ?
58. E : Monsieur ?
59. Ppr : Oui.
60. E : Est-ce qu'on peut… ? J'ai rentré ce chiffre au départ nombre comme résultat là et je fais ANS, ANS et ANS et ça fait le même résultat 3 fois ?
61. Ppr : …
62. E (une autre de même binôme) : On va essayer pour voir si ça marche.
63. Ppr : …Là, je vous laisse vous débrouiller…
64. Ppr : Là, vous êtes bien passés par la question suivante ? Pour l'exercice, pour la question 2 ?
65. E : Oui.
66. Ppr : Très bien. Vous oubliez pas de mettre les noms sur les feuilles… Claudine, tu voudrais pas mettre ton nom, sur la feuille ? (…) C'est quoi la question, Jérôme ?
67. E : Dans la question 2, le but c'est pas de faire le minimum de touches ?
68. Ppr : Qu'est ce qui est marqué dans l'énoncé ? …
69. E : Mais non, il faut noter.
70. Ppr : Simplement, vous noter le nombre de fois, ah ? (…)
71. Ppu : C'est bon ? Qui n'a pas fini ?
72. Es : Moi, moi.
73. Ppu : Allez, je vous laisse deux petites minutes et puis, on va passer, on va passer à la correction (…) C'est bon ? Vous avez noté le nombre de touches, le résultat ? (…)
74. Es : Ouais.
75. Ppr : Voilà, tu as tapé le nombre de touches, c'est parfait. Impeccable. Bon, je ramasserai ça. Je ramasse ça. On en parlera en classe entière, le but de cette manip., de l'exercice etc. Vous allez voir que ça a un lien avec l'exercice suivant. C'est bon, je peux ramasser ? …Merci. Ingrid et Karen, je peux ramasser.
76. Es : Non.
77. Ppr : Là, j'ai un…C'est. C'est bon Claudine ?
78. E : Non.

79. Ppr : Nombre de fois, vous avez essayé de compter un peu près le nombre de touches que vous avez tapé ? Merci.
80. E : Ca marche pas.
81. Ppr : Si tu reviens dessus, je pense que tu dois pouvoir. Si tu fais ça là. Non ?
82. E : Bah, non. Je me suis même trompé à la fin d'un chiffre que je sais pas comment. Je vais pas tout commencer ? Ah.
83. Ppr : Voilà. Je peux ? Non ? T'as pas compté encore. (à un autre binôme) C'est bon ?
84. E : Oui. C'est normal qu'on arrive pas au même résultat ?
85. Ppr : C'est pas normal. Vous avez le même calcul normalement…
86. E : Moi aussi, t'inquiètes...
87. Ppr : Merci.
88. E : Le dernier, t'as trouvé combien ?
89. E : 296
90. E : Oh, on a pas ça.
91. Ppr : T'as mis le nombre de touches ?
92. E : Ouais.
93. Ppr : Tu mets le résultat ?
94. E : C'est bon…
95. E : Etablissement ? J'ai mis LMG ?
96. Ppr : Pardon ? Oui, tu mets LMG, on sais ce que c'est. Merci. C'est bon, je peux ramasser Julie. Allez. Merci, merci. Tout le monde me l'a donné ? Merci.
97. Ppu : Voilà, alors, je vais marquer le résultat que vous avez trouvé au 1 et au 2.
98. Ppr : Tu mets un point d'interrogation, si tu, tu dis en gros comme ça ah ? Et tu me le donnes.
99. E : Si j'ai pas encore trouvé ?
100. Ppr : Tu mets, t'as trouvé le résultat ? Ah, non. Tu mets un point d'interrogation et puis…
101. E : J'ai pas mis comment trouver ?
102. Ppr : D'accord.
103. Ppu : Voilà, je vais marquer les résultats au tableau. Vous y retournez ? Alors, la réponse une que vous avez été très nombreux à trouver la bonne réponse. La réponse une c'est (il commence à écrit au tableau) 23,872762, normalement ah ? D'accord ? Pour la réponse 2, il y en a un petit peu de tout. Ah, donc, la bonne réponse, ça doit être ceci. Voilà, normalement c'est ceci.

Au tableau :

Réponse 1 = 23,872762
Réponse 2 = 296,888424

104. Ppu : Alors, ce que je vais faire, je vais demander à un groupe de passer ou du moins un élève de passer et à un de ses collègues de venir compter le nombre de touches en même temps pour vérifie qu'on trouve la même chose. Par exemple, Sandy tu peux venir et puis, je sais pas, soit Delphine, soit Elsa. Tu peux venir sur la calculatrice. Tu veux ta feuille ? Pour voir le calcul. Ca peut aider, et puis il y a une des deux pour compter le nombre de touches. Vous pouvez venir avec qu'elle et puis vous comptez le nombre de touches. Voilà, une des deux. Vous regardez bien ce qu'elle va faire. T'as trouvé combien de touches en tout Sandy ?
105. E : 64
106. Ppu : 64. Peut-être, donc vous comparez, vous comparez à ce que vous avez fait, vous.

Les réponses écrites :

Réponse 1 : 23,872762
Réponse 2 : 296,888424
Nombre de fois : 64

(La classe observe ce qu'elles ont fait sur la calculatrice publique)

107. Ppu : Regarde, regarde ce qu'elle fait, en même temps. Regarde ce qu'elle tape notamment. Tu peux voir tout ce qu'elle tape là (…) Claudine et Laura, suivez ce qu'elle fait.
108. Es : C'est pas Claudine.
109. Ppr : Oui, oui, j'ai bien dit c'est Claudine. Non, non, c'est Laura, pardon.

Le procédé de calcul

3,141759682 = 8 ; × Ans × Ans × Ans = ; + 6 × 3,141759682 × 3,141759682 = ; - 3 × 3,141759682 = ;

110. Ppu : Voilà, vous avez suivi. Combien de touches Elsa ? 65. (Et il le note au tableau)

Au tableau :

65 touches

111. Ppu : Elle a tapé 65 fois sur les touches. Est-ce que vous avez des questions, des commentaires pour l'instant ?
112. Es : …
113. Ppu : Vous avez vu la façon de procéder ? Voilà, je vous donne un nouvel exercice. C'est l'exercice 2. Voilà, sur la calculatrice, vous passez bien à l'exercice 2. Alors, simplement, vous vous mettez par binôme. Donc, Ingrid et Karen. Une des deux, vous fermer une calculatrice. Vous fermer que la calculatrice, ah ? Là, vous allez vous mettre en trinôme. Les autre binômes, c'est Manon et Claire. Manon et Claire ensemble, Julie et Maxime ensemble. Thibault et Jérôme c'est fait. Caroline et Claudine c'est fait, c'est ensemble. Laure et Marie ensemble. Et Samantha et Lauranne ensemble. Voilà. Vous fermez qu'une calculatrice ah ? Vous fermez qu'une calculatrice sur les deux. D'accord ? Claire avec, je sais plus avec qui. Avec Manon, je crois.
114. Ppr : Je vous donne trois feuilles, d'accord. Vous êtes passé l'exercice suivant ? Vous marquez vos noms sur les feuilles.
115. E : J'ai déjà marqué.
116. Ppr : Tu marques bien…

117.E : Nouvel exercice.
118.Ppr : Nouvel exercice. T'as fait déjà nouvel exercice numéro 2 ou pas ?
119.E : J'ai fait 2e question avant.
120.Ppr : T'as déjà fait. Fais nouvel exercice, 2.
121.E : J'ai déjà fait 2.
122.Ppr : D'accord, tu fais 2 bis, alors. 2 bis. (Il passe par chaque binôme pour distribuer les feuilles) OK. Voilà, vous marquez bien vos noms sur les feuilles. Vous avez une feuille d'instruction, une feuille navette et une feuille de brouillons. Une feuille d'exercice, une feuille navette, une feuille de brouillons. Vous marquez bien vos noms. Une feuille d'exercice, une feuille de brouillon, une feuille d'exercice, pardon…Exercice, navette et brouillon. Vous marquez vos noms. Exercice, navette, brouillon d'abord, pardon et navette. Vous marquez bien vos noms. Là, les trois filles, vous avez 3 feuilles : une feuille d'exercice, une feuille de navette et une feuille de brouillon. Vous marquez bien vos noms dessus.
123.E : D'accord.
124.Ppu : Voilà, vous regardez par là, deux minutes, s'il vous plaît. Deux minutes, Caroline. Donc, vous avez vu, vous avez une feuille d'énoncé. L'exercice est en trois phases. Dans la première phase, vous devez simplement écrire sur la feuille, vous avez 10 minutes pour la faire. Vous devez écrire sur la feuille ce qu'on doit taper sur la calculatrice pour obtenir le bon résultat. Mais vous marquez pas la formule ah ? Vous marquez simplement ce qui a à taper sur la calculatrice pour obtenir le bon résultat. Au bout de ces 10 minutes, que vous ayez finit ou non, je vous demanderai de transmettre votre feuille au binôme voisin. D'accord ? Donc que vous l'échangerez 4 par 4. Et là, vous aurez, chaque binôme aura 5 minutes pour essayer de faire le, essayer de suivre les instructions de ses collègues. Des questions ?
125.E :…
126.Ppu : Voilà, vous pouvez y aller. Vous avez 10 minutes pour écrire un message qui permet de faire le calcul.
127.Ppr : Regarde bien ta feuille d'énoncé.
128.E : Ah, oui. J'ai pas…
129.E : Il faut écrire le calcul, non ?
130.Ppr : Il faut écrire des instructions qui permettront à ton, à tes collègues de faire le même calcul et de trouver le bon résultat…(à un autre groupe) Vous êtes bien passé à l'exercice 2 ? D'accord ?
131.E : Oui.
132.Ppr : Mais je veux dire sur la calculatrice vous avez fait « séance », « exercice 2 ».
133.E : Ouais, ouais.
134.Ppr : D'accord. Donc, il faut, il y ce calcul à faire, ah ?
135.E : Celui là.
136.Ppr : Oui. Voilà. Et il va falloir donner des instructions. Ici il va falloir écrire un message pour dire à vos collègues comment effectuer le calcul sur la calculatrice. Qu'est-ce qu'ils ont à taper. Le but c'est qu'ils trouvent le même résultat, le bon résultat (…) (à un autre groupe) Voilà, le message, vous marquerez bien sur la feuille navette, d'accord ? Eux, ils marqueront bien leur nom et ils vous indiqueront ce qu'ils disent.
137.E : Brouillon, tu fais sur un brouillon.
138.Ppr : Vous pouvez utiliser un brouillon s'il vous voulez...(à un Ob.) Du point de vu de temps, c'est bien, en fait. C'est bien on a du temps en fait (…)
139.E : J'ai pas compris.
140.Ppr : Qu'est ce que…T'as lu l'énoncé ?
141.E : Ouais.
142.Ppr : Qu'est ce que t'en dis ? (Il relit l'énoncé) « Dans le message n'indiquez que les touches de la calculatrice. En suivant vos instructions, ce binôme doit obtenir la valeur numérique de z à l'écran de la calculatrice ». Donc vous allez donner exactement la succession de touches à taper pour que vos collègues obtiennent le bon résultat, pour le nombre de z.
143.E : Mais il connaît pas y.
144.E : Monsieur
145.Ppr : Thibault, regarde bien. (à une autre élève) Oui, non pardon là. Excusez-moi mais, ils vont pouvoir taper ça sur la calculatrice ?
146.E : Euh, ils tapent 2 fois…
147.Ppr : Essaie ! Le tape comment ça. Vous tapez les touches de la calculatrice ah ? Regardez « n'indiquez que les touches de la calculatrice ».
148.E : Il y a pas « z = » ? Je comprends rien.
149.Ppr : Est-ce qu'il y a une touche z sur la calculatrice ?
150.E : Mais non. Mais non.
151.Ppr : D'accord. Imaginez que vous avez ce calcul à faire, comment vous le feriez ?
152.E : Ah, d'accord (…) (A un autre groupe) Bon, vous en sortez ? Bof.
153.E : La valeur de z. Il faut qu'il trouve ça sur son écran ou il le retrouve en ayant remplacé…?
154.Ppr : Ah ton avis, à ton avis, z là, ça va être un nombre ou pas ?
155.E : Bah oui, ça va être un nombre par ce que…
156.Ppr : Ca va être un nombre. Donc, il faut qu'il trouve ce nombre.
157.E : Ah, d'accord. Et nous on fait le calcul avant ? Pour savoir si ça va être bon ?
158.Ppr : Ah, vous avez droit, vous avez droit de faire ce que vous voulez, ah. Simplement, il va falloir donner le message en suivant les instructions (…)
159.E : Monsieur.
160.Ppr : Oui.
161.E : On doit leur donner un calcul ?

162.Ppr : Vous devez … Regardez bien ce qui est écrit « Il dispose la même calculatrice que vous. Dans le message n'indiquez que les touches de la calculatrice ».
163.E : Donc, tout ce qu'on a à taper…
164.Ob. (à propos de la cassette) Est-ce que c'est finit la cassette ?
165.Ppr : Non, ça tourne encore. Je sais pas combien il en reste. Il en reste un tout petit peu. Ouais. Ce qui est intéressant c'est que, tu vois, il y pas tellement qui font des essais. Regarde autour de toi. Il y en a quelques uns qui essaient. Les autres, tout de suite, ils écrivent.
166.Ob. Ils écrivent des messages. Oui.
167.Ppr : Pardon ?
168.E : //la calculette
169.Ppr : Vous avez droit de faire ce que vous voulez. Vous devez à la fin, écrire un message. D'ici 5 minutes, il faut que votre message soit écrit...
170.E : Monsieur, il dit que le 2 a été déjà servi.
171.Ppr : Tu mets 2 bis…Parce que vous avez du passer…
172.E : Pour l'autre, j'ai mis 2 aussi.
173.Ppr : Voilà, c'était la question 2, ah. C'est pas grave. (à un autre groupe) Elle est où votre feuille d'énoncé.
174.E : Là, on mis 875. Je sais pas si c'est logique ?
175.E : Je sais pas. Fais voir. C'est un grand résultat en fait.
176.Ob. (à l'enseignant) L'absence de parenthèse ici amène ça et le problème, c'est qu'elle ne contrôle pas le résultat final. Elle n'a pas le moyen pour revenir sur cette faute là. Il y a pas de contradiction, donc il y pas de rétroaction en fait, donc qu'on est obligé, comme tu l'as fait, de faire appel à la règle.
177.Ppr : Ouais, ouais (…)
178.Ob. C'est intéressant. Elles ont tout //…
179.Ppu : Aller, on procède aux échanges…Donc, même si vous avez pas encore fini, c'est pas grave. Karen et Ingrid, vous allez passer aux voisins. Voilà, vous passez aux voisins. Vous passez la fiche navette aux voisins ou aux voisines. Allez, Marie et Laure, allez, vous passez telle que celle ah. Voilà. Alors, ceux qui reçoivent la fiche, marquent bien leur nom, dans la partie « phase 2 ». Sur la fiche « phase 2 », vous écrivez le nom de celui qui reçoit la fiche…
180.E : Là on a la fiche de Manon et on met…
181.Ppr : Là, vous notez votre nom. Binôme récepteur, c'est vous (…) (à un autre groupe) Qu'est-ce que vous faites là ?
182.E : On a pas encore fini.
183.E (une autre du groupe) : En fait on n'a pas recopié au propre donc je prends un brouillon et j'écrit là. Et là, j'écrits quoi là dedans.
184.Ppr : Voilà, tu suis les instructions. (à un autre groupe) Vous avez échangé ?
185.E : Ouais.
186.E : En fait ils ont échangé mais la fiche navette est resté là bas parce que ils ont pas encore…
187.Ppr : D'accord. (il prend la fiche et leur donne) Voilà. Donc là, vous marquez votre nom et vous écririez le résultat en suivant les instructions. Vous mettez votre nom, binôme récepteur.
188.E : Monsieur, on a pas le nombre de calcul ?
189.Ppu : Oui, vous avez pas les mêmes calculs, ah. Quand vous avez échangé vous avez pas les mêmes calculs, ah. Donc cherchez pas à avoir le même résultat…
190.E : Donc on n'a qu'à taper ce qu'ils ont écrit là. On suit ce qu'ils écrivent là.
191.Ppu : Vous tapez, voilà. Vous tapez ça et vous écrivez ce que dit la calculatrice.
192.E : Ah bon. Ah, seulement le résultat.
193.Ppr : C'est tout. C'est pas compliqué.
194.Ppu : Vous avez deux ou trois petites minutes pour faire ça.
195.E : Monsieur, on écrit là dessus ?
196.Ppr : Voilà, l'écran de la calculatrice.
197.E : Mais juste le résultat ou ?
198.Ppr : Vous avez suivi exactement les instructions ? Seulement les résultats final ah
199.E : Ah, oui le résultat final
200.Ppr : (à un autre groupe) C'est pas la même chose que vous ah ? C'est pas le même calcul que vous ah ? Vous tapez simplement ce qui est écrit.
201.E : On a mis ça.
202.Ppr : Et bah, il y a des erreurs. C'est pas grave. Vous marquez ça.
203.E : Regardez ça, monsieur. Elle a mis 2 virgule, 9 virgule, 1 virgule…257.
204.Ppr : Voilà, tu…tu interprètes ah…je sais pas…Normalement tu devrais taper exactement ce qui sur, ce qui est écrit ah.
205.E : Ouais.
206.Ppu : Si vous avez…si vous obtenez des choses du type « erreurs de syntaxes », « nombre invalides » ou je sais pas quoi, vous écrivez ce que vous voyez sur l'écran ah.
207.E : Et si ceux qui marchent pas // ?
208.Ppr : Vous attendez qu'ils aient terminé et vous échangerez.
209.Ppu : Allez, ça devrait pas être très long normalement (…)
210.Ppr : Vous avez fait, vous avez écrit ? Ce que vous avez vu à l'écran ? D'accord.
211.E : Oui.
212.E (du groupe) : Non, car c'est le résultat.
213.Ppr : Normalement, tu barres ça car t'en as pas besoin. (à un autre groupe) C'est…Tu mets que le résultat.
214.E : Ah, ce qu'on a calculé.
215.Ppr : Vous marquez ce que vous avez, le résultat final que vous avez sur la calculatrice, vous l'écrivez là.
216.E : On écrit ça en fait. Ou le résultat.

217.Ppr : C'est en bleu (…)
218.Ppu : C'est bon, vous avez fini ?
219.Es : Non, non.
220.Ppr : Samantha et Laura, vous avez fini ? Vous retournez votre fiche navette et vous avez euh, pas tout à fait 10 minutes, un peu moins de 10 minutes pour corriger votre message s'il y a besoin.
221.E : Quel message ?
222.Ppr : Et bah, votre message c'est ce que les instructions que vous avez donné.
223.E : C'est bon les filles ?
224.Ppr : Voilà, dès qu'elle ont fini vous…(à un autre groupe) vous avez noté le résultat ?
225.E : Non.
226.Ppr : Allez, notez le résultat. (à un autre groupe) vous avez noté le résultat ?
227.E : Oui.
228.Ppr : Oui. Vous leur redonnez, ah, avec leur feuille de brouillon. Voilà. Donc, bah. Vous faites en sorte de corriger votre, votre feuille ah.
229.E : Monsieur, nous on a trouvé pareil.
230.Ppr : Ouais.
231.E : Alors, on fait quoi ?
232.Ppr : Mais vous pouvez peut-être demander si c'est le bon résultat ?
233.E : Et comment on peut savoir ?
234.Ppr : Je sais pas. Cherchez (…) Comment est ce qu'on peut savoir si c'est le bon résultat ? (à un autre groupe) C'est bon ? Vous avez récupéré votre feuille ?
235.E : Ouais.
236.Ppr : Donc, vous êtes content de votre calcul.
237.E : Ouais. Mais on sais pas le résultat, à la base…
238.Ppr : On peut pas avoir une petite idée ?
239.E : Alors, là.
240.Ppr : Non ?
241.E : Non.
242.Ppr : Bah, posez vous la même question. Est-ce qu'on peut avoir une idée du résultat ? (…)
243.E : C'est pas possible, non.
244.E (du groupe) : Mais on peut pas. Y a des erreurs par tout. On sait pas si c'est ce calcul ou pas.
245.Ppr : C'est toujours la feuille d'instruction ça ?
246.E : Non c'est la leur.
247.Ppr : Oui, donc on vous l'a pas rendu encore ?
248.E : Non.
249.Ppr : Vous notez que le résultat final. Bon, on va noter ça. C'est bon, vous avez fini là. Allez. (à un autre groupe) Vous avez récupéré la votre ?
250.E : Non, elle est là.
251.Ppr : Allez.
252.E : Et un brouillon avec.
253.Ppr : Donc, vous vérifiez bien…
254.E : En fait va écrit ce qu'on a écrit.
255.Ppr : Vous vérifiez d'abord si est ce que votre message était clair, est qu'ils ont trouvé le bon résultat ? Et est-ce que ce que vous aviez prévu c'était bien ça, en fin ? D'accord ?
256.E : En fait, c'est presque. C'est près.
257.E (un autre du groupe) : à un près //Nous c'était clair//
258.Ppr : Ca c'est ce que vous aviez donné là ?
259.E : Non, c'est ça. Mais parce qu'on a pas eu le temps d'écrire //
260.Ppr : (il regarde la feuille) (…)
261.E : C'est pas clair monsieur ?
262.Ppr : Bah, d'accord. Je sais pas quoi. Est-ce que vous avez vérifié, est-ce que ça marche ce que vous faites.
263.E : Ah, on a fait, le résultat mais…//
264.Ppr : D'accord.
265.E (un autre groupe) : C'est bizarre, quand on fait ça on a 1, 7, 19 et ici on a ça //
266.Ppr : Alors qu'est ce qui vous surprend ?
267.E : Normalement ça devrait garder. Par ce que moins 17 avec 19 ça fait deux.
268.E : // Mais il y ça derrière.//
269.E (à l'ob. du groupe) : c'est compliqué quand même ce que vous nous aviez demandé…
270.Ppr : Est-ce que c'est la touche de la calculatrice car votre message ne doit comporter que les touches de la calculatrice ah.
271.E : Mais Ans c'est une touche de la calculatrice.
272.Ppr : Oui, mais là, ce que tu as écrit là.
273.E (une autre élève du groupe) : Mais c'est le résultat.
274.E : Mais non, on a pas trouvé celui là.
275.Ppr : Dans le message, n'indiquez que les touches de la calculatrice.
276.E : Mais c'est le résultat du calcul sinon on peut pas faire.
277.Ppr : Est-ce que c'est une touche de la calculatrice ?
278.E : 2 ?
279.Ppr : Ca là.
280.E : Mais non.
281.Ppr : Ca là, 2,88.
282.E : C'est le résultat.
283.Ppr : Donc c'est une touche de la calculatrice. Donc ça ne peut pas faire partie du message. Juste pour savoir, le 31,2 il vient d'où ? D'où ça.
284.E : Là…
285.Ppr : D'accord, d'accord. (à un autre groupe) C'est bon vous avez modifié quelques choses ?
286.E : Bah, non.
287.Ppr : Vous dites ah, rien est modifié etc. (à un autre groupe) vous avez modifié quelques choses ou pas ?
288.E : Ouais mais…
289.Ppr : OK. Allez (à un autre groupe) avez fini ?
290.Es : Non. Et je fais Sto A, ensuite je fais…
291.Ppr : (il observe le travail du groupe) Alors, vous marquez bien, voir le brouillon ah, pour la correction du message là. Vous marquez là.
292.E : Oui. Et après je fais…//

293.Ppu : Alors, à qui je vais demander. Je vais demander à Julie et Maxime de venir dans un premier temps. Allez il nous reste très peu de temps. Vous allez taper juste le début de ce que vous avez écrit. Vous venez par là. Ouais. Vous tapez exactement ce qu'il y a sur votre feuille ah.
294.E : Ca ?
295.Ppu : Ouais. Voilà. Vous regardez bien, s'il vous plaît, ce qui se passe sur l'écran. Donc, Maxime tape. Tu peux t'assoire ah, Julie. Merci. Maxime tape exactement son message. Ceux qui n'ont pas encore fini. Vous terminez en silence.

Le message de ce binôme :

Ans = 1,254
2 × 1,254 × Ans × Ans = 3,943870128 + 15 × 1,254 – 5 =

296.Ppr (au binôme au tableau) : 2e question. J'oublie. Merci.

Ce que tape Maxime :

1,254 =

297.Ppr : Stop c'est là que je m'arrête. Ce que vient faire Maxime « 1,254 = ». Est-ce que c'était sur la feuille ?
298.E : Non, ouais.
299.Ppu : C'était pas le feuille. D'accord. On est d'accord. Après, je vais demander à Manon ou Claire de venir faire leur calcul. Vous venez faire ça ? Voilà, 1,254 il l'a déjà fait ah.

Le message de ce binôme :

1,257 =
4 × Ans × Ans × Ans – 9 ×1,257 + 21 =
÷ 31,2 × 1,257 – 26,7184 =
Ans × Ans – 7 × Ans + 9 =

300.E : Je refais car c'est 1,257.
301.Ppu : Tu vas refaire. D'accord. Vous la suivez bien ce qui se passe, d'accord. Alors vous avez pas forcement les mêmes valeurs ah, mais votre calcul, la structure du calcul il est le même, ah. Elle est la même chez tout le monde, la structure de calcul. Même si vous avez pas exactement les mêmes chiffres.

Le procédé de calcul :

1,257 = ; 4 × Ans × Ans × Ans – 9 ×1,257 + 21 = ÷ 31,2 × 1,257 – 26,7184 =

302.Ppu : Voilà, tu t'arrêtes là. Tu stoppes là. Vous regardez, regardez bien. Il y a un problème à cet endroit là.
303.E : Il y a un moins.
304.Ppu : Non, c'est pas ça. Vous avez tous les mêmes calculs et je crois que vous avez tous fait un peu les mêmes erreurs.
305.E2 : Ah, oui. Parce qu'elle a fait 31,2 c'est ça non ?
306.Ppu : Non.
307.E : Ca va être que le 31 qui va être …
308.Ppu : Ca va être que le 31 qui…
309.E : C'est le 31 qui va être divisé et après l'espèce multiplié par …
310.E (au tableau) : Ah, oui. Il aurait du falloir calculer la ligne du dessus.
311.Ppu : Il aurait du falloir calculer la ligne du dessus pour faire, pour faire ?
312.E (au tableau) : La division.
313.Ppu : Pour faire la division, c'est-à-dire que là, vous regardez bien ça. On termine avec ça et on en parlera en classe entière. Là, la division est faite que par 31,2 fois 1,257. Le moins, il est à part. D'ailleurs le résultat, il est négatif, alors que si on y réfléchit le dénominateur il est positif avec le numérateur positif ça devrait donner un résultat
314.Es : Positif
315.Ppr : D'accord. Vous laissez les feuilles. Vous laissez les feuilles sur les tables. Vous fermez juste la calculatrice. Merci pour votre coopération. Voilà, laisse la feuille sur la table, on va, on va ramasser.

I.2. Protocole de la situation 1 avec groupe 1 de la classe 2nd F2

1. Ppr : Ca y est, non, non, c'est bon. Alors, il faut simplement que je place mes élèves.
2. Ppu : Donc, c'est juste qu'il y a deux groupes d'élèves qu'il faut que je place en premier. Donc c'est l'autre là. Jo…nathan et Stéphane, ils ne sont pas arrivés. Amandine et Françoise et…Eve.
3. E : Je m'appelle Françoise (elle rit).
4. Ppr : Voilà, Françoise non, Eve, tu viens là. Vous vous mettez là-bas, là où il y a les micros. D'accord ? Les autres, vous répartissez dans la salle. Vous laissez deux places là qui sont libres. Bonjour.
5. Ppu : Vous ne touchez pas les claviers pour l'instant. Je vais vous expliquer ce qu'on fait.
6. Ppr (à des garçons qui arrivent) : Bonjour, allez, voilà, installez-vous rapidement, allez, je vous ai gardé les deux meilleures places.
7. Es : Oh, non. On rêve ou quoi ?
8. Ppr : Allez, installez-vous là, vite (…)
9. Ppu : C'est bon alors ?
10. E : La mallette ?
11. Ppu : La mallette ? Ah, bah, voilà, elle est là. J'ai oublié juste de la remplir...Voilà, donc, la première chose, je vous présente Mme Bessot, M. Birebent et M. Nguyen qui sont, comme je vous l'avez dit, qui sont venu faire un travail de recherche. Donc, leur travail, c'est un travail de recherche sur l'enseignement des mathématiques et ils s'intéressent particulièrement au travail sur les calculatrices. Donc, on va vous proposer un certain

travail. La première chose qu'on va faire c'est de démarrer la calculatrice ensemble. Donc, vous avez tous normalement, sur votre écran, un raccourci vers calculatrice point bat. Vous démarrez ça, il y a un écran noir. Ensuite, il y a une petite fenêtre où on vous demande nom, prénom etc. Vous remplissez ça. Vous rentrez nom, prénom.

12. Ppr : Il faut que tu fermes la calculatrice car elle restait ouverte et tu recommences. Voilà.
13. Ppu : L'année scolaire, vous mettez 2003.
14. E : 2003 ?
15. Ppu : Ouais car c'était la rentrée 2003…Voilà. Je vous donne une feuille d'exercice.
16. Ppr : LMG que tu rentres ah ? C'est pas grave, ce poste ne marche pas. Il faut que tu l'éteignes donc tu vas à côté là-bas. Voilà.
17. E : Pourquoi 2003 ?
18. Ppr : Rentrée 2003.
19. Ppu : Voilà. Tout le monde a une feuille ? d'énoncés ? Donc, je vous explique bien. Vous regardez bien là. Svp, regardez bien par là. (Il ouvre l'ordinateur public) Donc, je rentre Seconde E. Vous êtes groupe B. C'est ça ? Donc seconde E, MLG, 2003. Voilà, vous avez tous une feuille sur laquelle il y a un exercice et 2 questions. Alors, pour pouvoir démarrer la calculatrice, il y deux choses à faire. La 1ère chose c'est d'allez en haut dans « séance ». En haut à gauche, dans le menu « séance », « nouvel exercice », vous rentrez le numéro de l'exercice 1. D'accord ? Vous ne rentrez pas un nouvel exercice tant que j'aie vous donné pas une autre feuille. Là, les deux questions à faire sont sur le même exercice. Il y a un 2e chose à faire, c'est de retourner dans « séance » et faire 1ère question. Quand vous passerez à la 2e question, et bah, vous faites « séance » et « 2e question » mais pas « nouvel exercice ». D'accord ? Vous répondez « Oui » à la question. Et puis, ensuite, il faut allumer la calculatrice en appuyant sur la touche « AC/On ». Voilà donc vous avez accès à la calculatrice, vous faites ce qu'on vous demande sur la feuille de consignes. D'accord ? Donc dans cette phase là, c'est chacun pour soi…Donc vous vous apercevez, c'est une calculatrice qui est, qui marche pas tout à fait comme les calculatrices normales puisqu'il y a certaines fonctions ne sont pas accessibles, d'accord ? Donc il faut que vous vous débrouilliez avec ce qu'on vous propose, ah ? C'est le but de la manip. Ca va Ophélya là ? (…) Vous marquerez bien votre nom en haut de la feuille. Votre nom et votre classe…
20. E : On peut pas effacer ?
21. Ppr : Alors avec quoi on efface d'habitude, Maud ? Regarde toutes les touches.
22. E : Il y a AC/Del.
23. E : C'est C/Off.
24. Ppr : Il y pas de parenthèses. Tout ce qui en est gris c'est pas accessible donc il faut débrouiller sans. Regarde bien ce qui est écrit Laura.
25. E : On doit trouver ça ?
26. Ppr : Je sais pas. Tu notes.
27. Ppu : Il faut, il vous faut, il vous faut un crayon ah ? Un crayon à papier ou un stylo plume.
28. E : Monsieur, après avoir fait la 1ère question on peut faire la 2e ?
29. Ppr : Voilà. Vous dites « séance » « question suivante ».
30. Ppu : Donc, il y a des touches qui ne sont pas accessibles ah ? Vous avez remarqué peut-être ? Il y en a qui disent que les parenthèses ne sont pas accessibles. Le carrée apparemment n'est pas accessible non plus.
31. Ppr : T'est pas obligé de sélectionner chaque fois ah ?...C'est vous qui choisissez (…) Oui, il y en a qui m'appelle, oui.
32. E: J'ai pas effacer ça avec…C'est pas grave ?
33. Ppr : Tu dois pouvoir effacer avec la touche AC non ?
34. E : AC et bah, oui (...)
35. Ppr : Vous en sortez ?
36. E : On peut pas…
37. E : On a pas besoin de passer à la question suivante, non ?
38. Ppr : Si, si, quand vous passez à l'exercice deux vous le mettez. C'est juste pour qu'on puisse se retrouver sur ce que…Question suivante, voilà, question suivante. C'est juste pour qu'on puisse se retrouver sur ce qui est tapé sur la calculatrice…
39. E : Ah, d'accord.
40. Es : Chaque fois il faut passer à la question suivante ?
41. Ppr : Quand vous passez à la question 2 ah, ? Simplement. Voilà.
42. E : Monsieur, on peut utiliser plusieurs fois //
43. Ppu : Vous avez droit à tout ce que vous voulez. Dans deux minutes, je ramasserai les copies (…) Ah, oui, voilà, il y a une chose importante.
44. E : Oh, non.
45. Ppu : il y a une chose importante, c'est, que j'ai oublié de le dire, désolé. C'est qu'il va falloir compter le nombre de touches que vous aurez frappé.
46. E : Non.
47. Ppu : Mais vous pouvez, vous pouvez l'estimer ça. Même si tu ne fais pas entièrement les calculs.
48. E : Non, je refais.
49. Ppr : Tu comptes un peu près. C'est pas l'unité près ah ? Ce qui compte c'est pas d'avoir l'unité près mais d'avoir l'ordre de grandeur.
50. Ppu : Vous avez vu, dans le 2e exercice. Il faut compter le nombre de touches que vous aurez tapé (…)
51. E : En fait, y a un moyen pour faire court mais on sais pas faire.
52. E : On peut pas, il y a ni ça ni les parenthèses.
53. E : Si, on peut faire.
54. Ppr : Réfléchissez…Une calculatrice, ce n'est pas une voiture, ah ? Vous faites ce que vous voulez avec, ah ?
55. E : On peut personnaliser ça ?

56. Ppr : Non, ça, vous pouvez pas. Mais vous pouvez, sur les touches, vous pouvez faire ce que vous voulez…
57. Ppu : Donc, vous pensez bien compter le nombre de touches, ah ?
58. E : Combien de clicks ça ?
59. Ppr : Pardon ?
60. E : Combien de clicks en fait.
61. Ppr : Voilà, combien de clicks (…)
62. Ob. : Qu'est-ce que c'est, ça ?
63. Ppr : C'est un bracelet que quelqu'un a perdu tout à l'heure.
64. Ob. : C'est Sandy qui a des bracelet comme ça.
65. Ppr : D'accord.
66. E : On doit arrondir ou pas ?
67. Ppr : Qu'est ce qui est dit ? …il y a rien est dit. Donc tu fais à ton idée.
68. E : Comment on fait pour revenir ?
69. E : On a pas de nombre différents mais pourquoi on a de différents résultats ?
70. Ppr : Bah, normalement, vous avez tout le même calcul ah ? Voilà, t'as compté le nombre de touches ? Je peux prendre ta feuille ? Je peux ramasser Caroline ?
71. E : Oh, il faut que je remplisse ?
72. Ppr : Je te fais perdre le fil ? Merci. Voilà, ce qui compte c'est l'ordre de grandeur à 2, 3 près. C'est bon Nicolas ? C'est bon Jonathan ? Tu as fini ?
73. E : Non
74. Ppr : Non, allez (…)
75. Ppr : C'est bon ? Le nombre de touches ? Jonathan, c'est bon, non ?
76. E : Non.
77. Ppr : Merci… C'est bon ? T'as une idée du nombre de touches tapées ou pas ?
78. E : Oui, mais c'est beaucoup.
79. Ppr : Beaucoup ? C'est bon, je peux ramasser ?
80. E : Monsieur.
81. Ppr : Ouais.
82. E : Comment je fais pour que //
83. Ppr : Ah, on le fait pas. C'est vrai que c'est un petit peu le problème mais on laisse comme ça. Voilà, ce qui compte c'est l'ordre de grandeur.
84. Ppu : Voilà. Je ramasse car il faut qu'on passe à la suite.
85. Ppr : Vous n'avez pas les résultats à me donner les filles ? Coline et Caroline ?
86. E : Je me suis trompé. J'ai oublié carré.
87. Ppr ; Bon, c'est pas grave. On va laisser comme ça. Tu marques simplement que tu t'es trompée.
88. E : On va filler le nombre. Si non, on peut faire comme ça, on a pas besoin d'aller dans l'exercice.
89. Ppr : C'est pas grave. C'est pas grave. Je vais ramasser.
90. E : T'a mis ton nom ?
91. Ppr : Voilà. Qui est-ce qui n'a pas encore mis son nom là ? C'est Caroline ? Elodie ? Laura. Tu mets ton nom et la classe. J'ai une autre feuille là, où il y pas le nom. C'est toi ? Non, c'est Ophélya ?
92. E : C'est moi.
93. Ppr : Tiens, mets ton nom et la classe…
94. Ppu : Voilà, alors je vais demander…Déjà, on va mettre peut-être les réponses sur les écrans. Mélodie, tu as fini ? Je peux prendre.
95. E : Ouais.
96. Ppr : Eve. Merci. C'est bon, tout le monde me l'a rendu ? Alors, les bonnes réponses, d'après vos feuilles. Pour les premiers calculs, il y a pas trop de. Vous regardez par là, s'il vous plaît. Jonathan, on arrête là, maintenant. Pour les premiers calculs, il y a un peu près une unanimité. Donc, si je note bien tous les chiffres qui sont donnés (il écrit le résultat au tableau), vous trouvez 23 virgule 872762. Pour le 2e, vous trouvé, pour ceux qui ont trouvé le bon résultat, vous trouvez deux cent quatre vingt seize virgule huit cents quatre vingt huit quatre cent vingt quatre.
Au tableau :
Réponse 1 = 23,872762
Réponse 2 = 296,888424

97. Ppu : Alors je vais demander à Elodie. Donc, vous êtes nombreux à trouver 80 touches, un peu près dans l'ordre de grandeur quatre vingt touches pour calculer ce résultat, le résultat de la question 2. Je demande d'Elodie qui a trouvé soixante, aux environs 60 touches de passer à l'ordinateur qui est ici et de venir nous montrer ce qu'elle a fait. Vous regardez bien. Oui, je te redonne ta fiche. Et en même temps qu'il y a Nicolas qui va venir contrôler le nombre de touches tapées. Tu viens, Nicolas, tu regardes ce qu'elle fait. Je vais donner ta fiche. Bon, en fait ce n'est pas très, très important mais.
Réponses écrites d'Elodie
Réponse 1 : 23,872762
Réponse 2 : 296,888424
Nombre de fois : ≅ 62

98. E : Non, c'est juste pour voir le calcul.
99. Ppu : Tiens, voilà. Vous regardez bien ce qu'elle a fait. Et ensuite, je vous distribuerai, on passera au 2e exercice. C'est l'exercice 2. Donc, je fais pas de commentaire, pas d'explication. Vous regardez comment elle a fait, ah ? (…) (la classe observe ce que fait Elodie)
100. Ppr : Nicolas, tu comptes en même temps ah ? (…) Voilà, tu arrives au même résultat, t'as trouvé combien ?
101. E : 64.
102. Ppr : Soixante quatre. Soixante quatre touches.
Au tableau :
≈ 60 touches
64 touches

103. Ppu : Vous avez vu ce qu'elle a fait ? Alors, il y en a qui a trouvé moins. C'est Stéphane ? Il y a, qu'il a mis quarante touches.

104.E2 : Mais alors là, je me suis trompé.
105.Ppu : Qu'est ce que tu en penses ? Vu la façon dont tu a fait, il y a des chances que t'as mis, t'as passé plus de de touches que soixante, ah ? On est d'accord. Voilà, on va s'en tenir là pour le 1er exercice.
Le procédé d'Elodie :
3,141759682 = ; Ans × Ans × Ans = ; × 8 + 6 × 3,141759682 × 3,141759682 – 3,141759682 – 1 ;

106.Ppu : Je vais vous donner un 2e exercice. Mais il est à faire en binôme.
107.E : C'est-à-dire ?
108.Ppu : Alors les binômes, je les donne, il y a, Stéphane et Jonathan. C'est les binômes habituels un peu près ah ? Ensuite, Johanna et Caroline. Donc, vous fermez une des deux calculatrices. Ah, pareil pour Stéphane et Jonathan. Et sur l'autre, vous passez à l'exercice numéro 2. Alors, qu'est ce qu'il y a d'autres ? Il y a Laura et Maude, vous mettez ensemble. Vous vous mettez ensemble, Laura, où elle est, elle est là-bas. S'il vous plaît. Il y a un trinôme, c'est Caroline, Mélodie et Elodie. D'accord ? Alors vous vous mettez vite ensemble, qu'on perde pas trop de temps. Caroline Portier, Mélodie et Elodie. Venez là. Mélodie, tu veux bien venir là. Caroline et Coline. Amandine et Eve, c'est fait. Et Aurélie Gladys, vous vous mettez ensemble. Vous fermez une et vous ne gardez qu'une calculatrice. Vous pensez à passer à l'exercice 2 dans la calculatrice, ah ? Vous dites bien, nouvel exercice, numéro 2. //
109.E : Nouvelle question ?
110.Ppr : Oui, voilà, première question. En fait, il y a une seule question dedans.
111.Ppu : Il y a qu'une question mais c'est un petit peu plus longue. Donc, je vous passe trois feuilles. Je vous passe une feuille d'énoncé, une feuille navette et une feuille brouillon. L'énoncé, vous marquez, vous marquez, votre nom dessus. La fiche navette, vous marquez aussi votre nom. Alors, ça se passe comment ? Donc, par exemple, si je prends deux groupes là. Ils ont des énoncés différents. Ils vont devoir faire un travail et l'échanger ensuite. D'accord ? Mais vous avez un énoncé qui est légèrement différent. Pensez pas que vous avez le même travail que votre voisin. Je vous distribue les feuilles, vous allez mieux comprendre. Donc, vous marquez bien le nom sur chaque feuille (…) Voilà. Tout le monde doit avoir 3 feuilles, normalement à la fin. Voilà, vous marquez bien votre nom. Est-ce que tout le monde a bien la fiche navette ? Car il m'en reste une. Oui, tout le monde l'a ? Vous l'avez pas ? Pardon ?
112.E : On a pas.
113.Ppr : Ah, voilà, c'est ça. Vous n'avez pas la fiche navette ?
114.E : Monsieur, c'est normale que j'ai pas…
115.Ppu : Ah, il y en a un petit…Non c'est pas normal. Donc il m'en faut, il m'en faut deux. Voilà, un petit problème de tirage. Alors, j'insiste sur une chose. Vous avez lu dans les instructions dans le message. Alors, je vous laisse lire le message et ensuite, ensuite je vous donne une petite précision. Lisez bien le message.
116.Ppr : Ca, c'est juste un brouillon ah ? Le message, il est à mettre là, ah ? D'accord.
117.E : C'est à marquer là dedans ?
118.Ppr : Voilà. C'est à marquer…Non, c'est pareil. T'as un problème. Il y a un problème. Vous avez la bonne fiche navette ? Non ? Vous avez pas la bonne fiche navette non plus. Ouais, merci.
119.Ppr : Voilà, un bon petit calcul. Il va falloir que vous expliquiez et uniquement avec les touches de la calculatrice, comment le faire et vous écrivez là. D'accord ? Vous avez un petit 10 minutes.
120.Es : En parlant ce qu'il faut faire ?
121.Ppr : Qu'est ce qui est écrit ?
122.Ppu : Alors, qu'est ce qui est écrit dans l'énoncé ? On vous dit que le message ne doit contenir que des touches de la calculatrice. D'accord ? Donc vous avez rien d'autre à mettre. Vous avez à mettre uniquement une succession de touches à appliquer pour indiquer comment faire le calcul et arriver au bon résultat. Ce que vous tapez sur l'écran. Ce que vous tapez ça. D'accord ? Il y a rien d'autre à ajouter. Vous avez du brouillon. Vous avez une calculatrice sous la main. Donc, vous pouvez faire ce que vous voulez (…)
123.E : y c'est quoi ?
124.Ppr : Regardez bien dans l'énoncé (…)
125.E : Monsieur ?
126.Ppr : Oui.
127.E : On a pas bien compris. Dans le message on met quoi, n'importe quoi ?
128.Ppr : Qu'est ce qui est écrit. Là, regardez « dans le message n'indiquer que des touches » « En suivant vos instructions, ce binôme » c'est à côté, ah ? C'est Eve et Amandine, ah ? « elles doivent obtenir la valeur numérique de z à l'écran de la calculatrice ».
129.E : C'est-à-dire je peut taper ça et ça ? On peut écrire ce qu'on voit.
130.Ppr : Ah, donc il faut indiquer comment faire les calculs. Quelle est la succession de touches à taper pour faire les calculs.
131.E : D'accord.
132.Ppr : Imaginez que vous avez ce calcul à faire. Comment vous le faites, vous ?
133.E : On calcule ça et après on rentre ça//
134.Ppr : Donc, le calcul que vous allez faire, vous l'indiquez à vos destinataire…
135.E : D'accord.
136.Ppr : (à un autre) y, vous ne connaissez pas, c'est ça ? Regardez bien l'énoncé.
137.E : Ah, si.
138.Ppu : Voilà, dans 5 minutes, on fera l'échange. Dans 5 minutes, on fera l'échange. Même si vous ne pensez pas l'avoir terminé complètement (…)

139.Ppr : Donc, le message que vous devez passer c'est quoi ?
140.E : C'est ça !
141.Ppr : C'est quoi ? C'est le résultat ?
142.E : Bah, oui.
143.Ppr : Quel est l'intérêt de passer le résultat ?
144.E : On passe les touches ce qu'on doit taper des touches pour arriver au résultat. Il y en a un paquet quoi ?
145.Ppr : Oui donc il faut se dépêcher un peu (à Ob. qui lui fait signe de tourner la cassette) j'ai tourné même s'il n'est pas encore fini. Je n'ai pas rembobiné ah, l'enregistrement sera pas au début. C'est pas grave ? (…)
146.E : On doit donner tous les calculs ? Ou juste les instructions ? //
147.Ppr : Ils ont pas le même calcul que vous
148.E : Ouais.
149.Ppr : D'accord ? Donc il faut leur donner une succession. Regardez ce qui est écrit dans l'énoncé. Il faut indiquer les touches de la calculatrice qui sont à utiliser pour trouver, pour que le binôme doive obtenir… la valeur numérique à la fin.
150.E : Quand on a à écrire 1, 4 il faut écrire comme ça. La touche 1, la touche point etc. ?
151.Ppr : Vous n'allez pas l'encadrer chaque fois. Pour la touche 1, vous écrivez 1, ah ? Car sinon, vous en avez pour des heures.
152.E : Mais tape là aussi, 1 point 257.
153.E : Mais non, il faut écrire à chaque fois la touche.
154.Ppr : Non, tu tapes 1, point…
155.E : Non, tu tapes directe là.
156.Ppr : C'est 1 point machin et ça correspond à 1a touche 1 et la touche point. D'accord ? Sinon, t'en as pour un petit moment là.
157.Ppu : Allez dans 2 minutes, on échange les feuilles.
158.E : J'ai fait une erreur et j'ai du tout recommencer.
159.Ppr : Vous commencez à écrire votre message…par ce qu'il faut qu'on échange les messages. Dans deux minutes je sera obligé de faire l'échange même si vous n'avez pas encore fini (…)
160.Ppu : Voilà, allez, vous marquez le message, vous marquez vite le message. C'est bon ? Commencez à marquez le message. Allez, dans une minute, on fait les échanges ah ? Donc, commencez à marquer le message.
161.Ppr : Normalement, ça, vous devez pas le mettre vous voyez ?
162.E : Ouais.
Le message de ce binôme :

① 1,254 = Ans × Ans × Ans × 2 + 15 × 1,254 – 5 =
② 32,1 × 1,254 – 7534 =
③ $\frac{17,75387013}{Ans}$ =
④ Ans × Ans – 5 × Ans + 7 = z

163.Ppr : Puisque dans le message il faut n'indiquez que les touches de la calculatrice. Eventuellement, vous pouvez garder ça pour vous mais normalement il faut pas l'écrire.
164.E : Monsieur, on peut écrire le résultat ?
165.Ppr : Est-ce c'est écrit de marquer le résultat ?
166.E : Non.
167.Ppr : Ah, ça, est-ce que c'est les touches de la calculatrice ça ? Vous pouvez les garder pour vous si vous voulez ah ?
168.E : Mais eux, ils peuvent pas écrire.
169.Ppr : Ils peuvent pas écrire ?
170.E : Puis que ces nombres. On a marqué à côté puis on a refait un calcul.
171.Ppr : Oui, mais ça, c'est votre brouillon et ensuite vous donnez ce qu'il y a à taper.
172.E : Mais s'il y a quelques choses à noter à côté ? Ou à retenir ?
173.Ppr : Vous avez donné. Vous ne pouvez donner que les touches de la calculatrice.
174.Ob. (à l'enseignant) : Ils cherchent à utiliser les mémoires depuis le début mais ils ont pas trouvé la solution.
175.E : A qui on donne ça ?
176.Ppr : Vous allez donner à votre voisin.
177.Ppu : Allez on fait l'échange même si vous n'avez pas totalement fini. On fait l'échange. Donc, Jonathan, Stéphane avec leur voisine. Nicolas, vous passez à Gladys et Aurélie. Tous les quatre, vous échangé entre vous. Voilà. Et puis les deux derniers groupes. Maude et Laura vous passer à Eve et à Amandine. Celui qui, ceux qui reçoivent la feuille marquent bien nom du binôme récepteur. Vous voyez, il y a un trait, il faut marquer votre nom, ah ?
178.E : Il faut marquer le résultat ou pas ?
179.Ppr : Pardon ?
180.E : Il faut marquer le résultat ou pas ?
181.Ppr : Non, non. Regardez bien ce qui est écrit. « N'indiquer que les touches de la calculatrice »
182.E : On a mis les points d'interrogation pour les résultats.
Le message de ce binôme :

4 × 1,257 × 1,257 × 1,257 – 9 × 1,257 + 21 =
Ans ÷ 31,2 × 1,257 – 26,7184 = ?
? × ? – 7 × ? + 9

183.Ppr : OK, allez. Est-ce que le point d'interrogation c'est la touche de la calculatrice ?
184.E : On est des cons.
185.Ppr : C'est pas grave. Vous passez comme ça. Allez. Voilà.
186.E : Ils vont comprendre.
187.Ppr : Johanna, tu marques bien le nom de, le nom qui est sur la calculatrice, en fait. Là, c'est pareil. C'est bon, vous passez votre feuille aux autres. Vous marquerez bien votre nom sur la feuille que vous allez recevoir. Voilà, vous marquez bien votre nom, là. Vous suivez les instructions et à

la fin, vous écrivez le résultat final, là. D'accord. Vous écrivez bien votre nom. Voilà.
188.Ppu : Si le résultat final c'est « erreur de syntaxe » ou « nombre non valide » etc., ou ceci cela, vous écrivez ce que vous voyez.
189.E : Et nous, on a un petit problème ce qu'ils ont vraiment pas fini. Ils ont juste donner le nombre.
190.E : Bah, ouais. On peut pas avoir le résultat.
191.Ppr : Vous faite ce qu'ils ont tapé. Vous marquez…On vous a donné la fiche navette.
192.E : Ouais, ouais.
193.Ppr : Vous avez marqué votre nom dessus ?
194.E : Ouais.
195.E : On a près que rien à remarquer ?
196.Ppr : Ca fait partie de l'exercice. Ne vous inquiétez pas.
197.Ppu : Voilà, vous avez deux ou trois minutes.
198.Ppr : C'est bon, vous avez fini ou pas ?
199.E : On cherche…
200.Ppr : Vous avez rien à chercher normalement, là. Vous avez juste à taper ce qui vous a donné.
201.E : Ca ?
202.Ppr : Vous avez juste à taper ce qu'ils vous ont donné.
203.E : Et bah, voilà.
204.Ppr : Aurélie, ce que vous avez écrit, cela ne contient pas que des touches de la calculatrice. Vous avez donné plus de chose là.
Message d'Aurélie :

Résoudre $y = \frac{2 \times 1,254 \times 1,254 \times 1,254 + 15 \times 1,254 - 5}{32,1 \times 1,254 - 27,7534}$

$z = y \times y - 5 \times y + 7$

↙

Résultat de y

205.Ppr : C'est juste, normalement des successions de touches à taper, ah ? (à un autre) Et voilà, et vous suivez les instructions qu'elles ont vous donné (…)
206.E : Mais oui, il y pas de touches, c'est pas simple.
207.Ppr : Il n'y a pas de touches ? Si si, il y en a une là-bas…
208.E : D'accord. C'est laquelle ?
209.Ppr : C'est celle-là. Vous écrivez le résultat final. Oui, voilà, vous rendez la feuille à Gladys et Aurélie. C'est bon ? Voilà, vous retournez la feuille. Vous avez fini ? Joana, non Jonathan et Stéphane, vous rendez la feuille et vous pouvez, vous pouvez maintenant bah, revoir votre message pour essayer d'en écrire un valable… jusqu'à la fin.
210.E : Et si c'est bon ?
211.Ppr : Euh, vérifiez, vérifiez que vous avez un bon calcul ?
212.E : Ouais.
213.Ppr : Vous avez tout vérifié ? Vous êtes sûrs de votre coup ?
214.E : Ouais.
215.E : Non.
216.E : Si, on avait marqué le résultat donc c'est bon. On a le même résultat quoi.
217.Ppr : Ca veut dire qu'ils ont trouvé la même chose que vous ? Mais est-ce que ce que vous avez trouvé est bon ?
218.E : Oh, non !
219.Ppr : Je sais pas ! C'est une question que je me pose.
220.E : Mais si, j'en suis sûre car on a commencé par faire le dessus, prendre le dessous puis diviser. Alors on recommence depuis le début (…)
221.Ppr : Nicolas, elles se débrouillent et si elles arrivent pas, ce n'est pas grave.
222.E : (en riant) Si elles arrivent pas c'est ta faute.
223.Ppr : Allez, si elles y arrivent pas c'est que votre message n'était pas suffisamment clair pour suivre les instructions. Le but c'est de donner vraiment ce qu'il y a dessus. Voilà, vous marquez à la fin le résultat et vous rendez la feuille. Ca y est ? Vous avez fait ?
224.E : On a trouvé ah. Monsieur on fait sur la calculatrice, on marque le résultat ?
225.Ppr : Voilà…Vous avez fini ? Ouais ? D'accord, on va leur donner ça. Temps qu'ils marquent leur nom. C'est juste le résultat final ah, qu'il faut écrire. Allez, je vais mettre ça là. Il faut bien mettre votre nom.
226.E : Monsieur, comment on sait sur le résultat qu'on a trouvé ?
227.Ppr : Vous savez pas. (à un autre) C'est bon, vous avez fini. Allez. C'est ça qui compte là.
228.Ppu : Voilà, vous avez cinq, cinq gros minutes pour réviser votre message, le corriger. Vous ne le redonnez pas au voisin, ah…Allez.
229.Ppr : Vous l'avez recommencé combien de fois ? (…)
230.E : 3 fois.
231.Ppr : Alors, si vous pensez aussi avoir totalement terminé, ce que vous allez fait, c'est d'essayer de faire le plus court possible. D'accord ?
232.Ob. : Ils ont été bloqué par l'usage syntaxique de mémoire donc je leur ai indiqué la syntaxe.
233.Ppr : Ouais, ouais. De toute façon c'est…(à un autre groupe) Vous avez réduit sensiblement ? D'accord.
234.E : Le reste je crois qu'on peut pas raccourcir car on a déjà le résultat. Donc ça ne fera pas le bon résultat quoi.
235.Ppr : Vous avez fait le tour de touches qui peuvent être utilisé ? (…) C'est bon.
236.E : Maintenant il faut simplifier.
237.Ppr : Soi le simplifier, soit le corriger. Vérifier que ce que vous avez donner ça marche que…
238.E : Mais on sais pas en fait le vrai résultat.
239.Ppr : Ouais. Ouais.
240.E : On peut refaire ça.
241.Ppr : Oui mais il faut pas redonner sous cette forme là. Il faut le donner sous forme d'une succession de touches.

242.Ppu : Alors, on a très peu de temps pour corriger là.
243.Ppr : Ca y est ? Vous avez récupéré votre feuille ?
244.E : On a pas encore le résultat fini là ?
245.Ppr : Alors essaie de voir s'il y a des choses à corriger, à revoir ? Et éventuellement, comment vous pouvez raccourcir votre message (…)
246.Ppr : C'est quoi là, - 5 ? Je peux voir ce que vous faites ? Il me faut la feuille en même temps. Alors, mais ça ce n'est pas de touches de la calculatrice, ça. Il faut retaper là.
247.E : Oui, mais c'est ce qu'on trouve, la, à la fin juste le résultat.
248.Ppr : Oui, mais bon. Normalement le message, il est comme ça.
249.Ppu : Bon, on va, on va arrêter là.
250.Ppr : Ca ce n'est pas une touche de la calculatrice. On ne peut pas taper ça sur une touche de la calculatrice.
251.Ppu : Bien, Caroline, tu veux bien passer au, à l'ordinateur. Vous pouvez passer tout les deux si vous voulez, nous montrer ce que vous avez fait. Je crois que j'ai pas passé l'exercice 2. Pardon ? Oui, oui, vous prenez votre message, pour venir le taper. Allez vite par là. Les autres, vous arrêter d'écrire. Vous regardez simplement ce que font Johanna et Caroline. Nicolas et Ophélya, vous regardez aussi par là. Stéphane. Alors, j'ai déjà la 1ère remarque c'est quelle utilisent la touche Ans (la classe observe le binôme) (…)

Le message de ce binôme :

① $1{,}254 =$
$\text{Ans} \times \text{Ans} \times \text{Ans} =$
$\text{Ans} \times 2 =$
$Ans + 5 \times 1{,}254 - 5 =$
$\frac{Ans}{32{,}1 \times 1{,}254} - 27{,}7534 =$

$\text{Ans} \times \text{Ans} =$
$\text{Ans} - 5 \times ① =$
$\text{Ans} + 7 =$

Leur procédé :
1,254 = ;
Ans × Ans × Ans = ; Ans × 2 ; Ans + 5 × 1,254 – 5 = ;
Ans ÷ 32,1 × 1,254 – 27,7534 = ;

252.Ppu : Voilà, vous allez arrêter là. Stop. Qu'est ce que vous en pensez là. Je crois qu'il y a de petits soucis avant, en fait sur l'utilisation de la mémoire mais comme c'est passé trop vite. On ne va pas pouvoir revenir là-dessus. Mais là, sur ce qu'elles ont fait. Vous avez pas ces valeurs là. Vous avez pas forcement exactement ces valeurs là mais sur chacune de vos consigne, la structure est la même et là, je pense qu'il y a quelque chose à dire.
253.E : C'est pas bon.
254.Ppu : Pourquoi c'est pas bon ?
255.E : Parce que là ils veulent diviser tout après Ans…
256.Ppu : Ouais. Ils veulent diviser le résultat qu'ils ont trouvé avant…
257.E : Par toute la ligne après.
258.Ppu : Par tout ce qui est derrière, normalement.
259.E : Et là ça fait 32 virgule 2 divisé par ça.
260.Ppu : Vous vouliez diviser par tout ça, là. Et ça, quand on tape tout ça, on divise uniquement…
261.E : 32 virgule 1.
262.Ppu : Par 32,1. D'accord ? Alors, le problème, il vient d'où. Il vient du fait que sur la calculatrice qu'on vous a donnée, il y a pas de…
263.E : Parenthèse.
264.Ppu : Y a pas de parenthèses donc qu'il aurait fallu trouvé un moyen. Je crois qu'il y en a qui ont trouvé. Trouver un moyen pour contourner cette absence de parenthèse là. Est-ce qu'il y en qui peuvent donner. Oui, Elodie.
265.E : On a déjà fait le calcul qui est au dessus quoi et après…
266.Ppu : Vous avez fait à l'avance le calcul qui est en dessus.
267.E : Et après on a noté le calcul…
268.Ppu : Donc, vous avez calculé le dessous, vous l'avé le noté.
269.E : Non, le dessous on l'a pas noté. On a pas noté le dessus. On a déjà calculé le dessus, on a noté le résultat.
270.Ppu : Donc, le dessus, vous avez calculé, vous l'avez noté.
271.E : Le dessous on a calculé et on fait dessus divisé par Ans
272.Ppu : Le dessous, vous l'avez calculé, d'accord ? Vous avez fait égale. Vous avez tapé le dessus qui était noté, divisé par Ans. Vous avez contourné l'absence de parenthèse en utilisant la mémoire Ans. D'accord. Alors, on va voir en classe entière d'une autre façon plus rapide en utilisant les mémoires pour faire ce calcul sans utiliser du tout, les parenthèses.
Le message de ce binôme
① 1,254 Sto A = A × A × A × 2 + 15 × A – 5 =
② 32,1 × A – 27,7534 =
③ $\frac{17{,}7538}{Ans}$ =
④ Ans × Ans – 5 × Ans + 7 = z

273.E : On a fait.
274.Ppu : Pardon. Vous avez fait à la fin ? Au début, qu'est-ce que vous avez fait.
275.E : On a noté 1,254 en mémoire comme A
276.Ppu : C'est-à-dire vous avez fait comment ?
277.E : 1 virgule deux cent cinquante quatre Sto A.
278.Ppu : Pourquoi vous n'avez pas utilisé plus après, alors ?
279.E://
280.Ppu : Donc ça c'était à la fin que vous l'avez fait. Pas au début.

281. E://
282. Ppu : D'accord. Donc vous voyez, éventuellement, on peut peut-être utilisé les lettres A, B, C, les mémoires A, B, C pour aussi raccourcir le message, le calcul à faire. Bon, on verra ça en classe entière. On essaie de le faire correctement car on garde la calculatrice on peut refaire tranquillement. Alors ce que je vous demande, c'est fermer les calculatrices. Vous fermez juste la calculatrice. Vous fermez que la calculatrice. Le reste, vous laissez sur les tables. Laissez l'ordinateur, sinon. Les feuilles, vous les laissez sur les tables, on va les ramasser. Au revoir.

I.3. Phase de synthèse du calcul 3 de l'enseignant en classe entière (2nd F2)

1. P : Et on arrête les discussions ? Donc, on va revenir, euh, on va revenir sur le, la séance de vendredi dernier pour faire quelques, quelques petites remarques. Alors, une première remarque que je voulais vous faire sur le, sur l'utilisation de la calculatrice. L'utilisation de la calculatrice qui vous avait été fournie. (À un élève qui lui donne un papier) Merci. C'est que j'avais remarqué, c'est que vous faisiez très peu d'essais. La calculatrice, vous ne saviez pas énormément utiliser. Vous aviez des calculs compliqués à faire et euh, vous osez pas faire d'essais, pratiquement. Par exemple, certains ont commencé à vouloir utiliser la mémoire mais savaient pas trop comment ça marchait. Mais il y avait pas eu, vous avez pas fait testé, vous avez pas fait d'expérimentation ou pratiquement pas. Je sais pas si vous avez pas, vous osiez pas peut-être, abîmer la machine, bon elle risque rien. C'est un logiciel, c'est comme votre calculatrice, ça risque absolument rien. Vous pouvez faire tous les essais comme vous voulez. C'est pas une voiture. Vous risquez pas de, de l'amener dans le mur. Donc, c'est peut-être une habitude à apprendre. Mais ça c'est vrai autant sur une calculatrice que sur papier. Il faut pas hésiter à faire des essais, à faire des test. Ah, vous vous ne souvenez plus d'une règle de calcul. Bas, vous faites un petit test avec 2 ou 3 nombres. Par exemple, sur les fractions, souvent vous avez des doutes, sur des règles de calcul, vous pouvez faire des essais. Sur une calculatrice, c'est pareil, vous pouvez faire tout tas d'essais. Une 2e chose que j'ai remarquée, qu'il y avait certains élèves qui trouvaient des résultats, euh…pas invraisemblables mais disons un petit peu étonnants vu les nombres qui vous étaient donnés. Vous avez trouvé certains résultats qui étaient un petit peu étonnants. Et il y a pratiquement, il y a personne, pratiquement parmi vous, euh qui a essayé de, comment, enfin de calculé un ordre de grandeur de ce qu'on lui demandait. Je prends un exemple. Je sais pas si vous avez encore cette fiche-là, là.
2. Es : Non
3. P : Non, vous les avez plus parce que vous les avez rendus. Donc je prends la question numéro 2. Le calcul de la question numéro 2. Dans la question numéro 2, (il écrit au tableau en même temps), on vous demandait, calculer 8 x au cube plus 6 x carré moins 3 x moins 1 avec x égal 3,141759682.

Au tableau :

Calculer $8x^3 + 6x^2 - 3x - 1$
avec $x = 3{,}141759682$

4. Alors bien sûr, c'est calculer la valeur exacte de ça, de tête ou à la main, c'est, c'est pas du tout évident. (Il fait signe à une élève qui bavarde) Laura. Par contre, ce qu'on peut, peut-être faire, c'est, c'est d'essayer d'évaluer un ordre de grandeur. Essayez de prévoir, environs le résultat qu'on va obtenir. Ca nous donnera, peut-être une idée de savoir si ce qu'on tape de la calculatrice c'est, contient une erreur ou non ? Alors, comment est ce qu'on pourra faire ça ? Jonathan ?
5. E : Avec 3.
6. Jonathan :// On prend 3 à la place de//
7. P : Peut-être, on va avec 3 alors. Est-ce que c'est très difficile de faire ça avec 3 ou pas ? Allez, on t'écoute Jonathan.
8. Jonathan : Bas, 8 multiplié par 3, 3 fois 3 fois 3.
9. P : Ouais, donc ça va faire combien ça ?

Au tableau :

Si $x \approx 3$
8×27

10. Jonathan : //
11. P : Jessika ?
12. Jessika : //
13. P : Donc, on a 8 fois 27 ah ? 3 au carré c'est 27. Ca peut-être même se simplifier. Peut-être que 8 fois 27 ce n'est pas très pratique. On fera mieux de faire, 10 fois 30 ou 10 fois ? Si vous augmentez celui-là, vous pouvez peut-être un petit peu…
14. Es: //
15. P : Voilà, diminuer un petit peu celui-là. Ensuite…
16. E : 6 fois 9.
17. P : Plus 6 fois 9. 6 fois 9 ça devrait pouvoir se faire. C'est combien ça, 6 fois 9 ?
18. E : 54.
19. P : 54, 54. Ensuite, 3 fois 3, neuf. Voilà. Bon, ça va combien, ça un peu près ? À la louche, ah ?

Au tableau :

$\approx 10 \times 25 + 54 - 9 - 1$
≈ 300

20. E: //2 cents…
21. P : Voilà 300, 290, 300 ah ? On sait qu'on va trouver quelque chose qui est proche de 300. Peut-être, peut-être, un petit peu en dessous euh. Ca c'est peut-être un peu surestimé, voilà. D'accord. Bon. On a, on a quand même une idée de, de l'ordre de grandeur qu'on doit trouver. Si on trouve des choses en10 puissance 5, par exemple. On va peut-être poser des questions. D'accord ? Voilà,

simplement c'est, c'est pour dire que, euh, on n'est capable de donner la valeur exacte, même peut-être à 10, à 10 ou à 20 près ou même à 50 près on n'est pas capable de donner la valeur exacte mais, on peut, peut-être quand même faire une évaluation. Je prends l'autre, l'autre expression. C'était un peu plus compliqué. Si je la retrouve ? (Il cherche dans ses papiers) (…) Donc, dans l'autre expression, alors, vous aviez pas tous les mêmes ah ? Puisque, il y en avait deux ah, il y en avait deux type de calculs. Donc, on vous demandait de calculer z (il écrit au tableau) égal y carré mois 7 y pus 9 avec y était un calcul fort sympathique que vous avez bien aimé. 4 x cube moins 9 x plus 21 sur 31,2 de x moins 26,7824 et avec x égal 1,257.

Au tableau :

$z = y^2 - 7y + 9$

$$y = \frac{4x^3 - 9x + 21}{31,2x - 26,7184}$$

avec x = 1,257

22. P : Voilà, est-ce que vous pouvez nous faire un petit essai là ? Je vous laisse faire à la main, vous sortez un petit brouillon et puis vous. Rapidement sans faire le, sans poser les opérations. Juste avoir, d'essayer d'avoir une petite estimation de l'ordre de grandeur. De tête, essayez, l'ordre de grandeur d'y déjà. Et ensuite, l'ordre de grandeur de z. En deux temps. Un virgule quelque chose, il faut, il faut le remplacer par quoi ?
23. Es : Par 1.
24. P : 1, ouais, pourquoi pas, ouais. (…) Tu dis combien ?
25. E : //
26. P : Environ 4 ? Tu peux nous donner la fraction que tu as obtenue ?
27. Es://
28. P : Environ 16 par 4 ?
29. Es : Ouais
30. P : Environ 4. On fait une grosse erreur si on remplace 1,257 par 1 ?

Au tableau :

$$y \approx \frac{16}{4} \approx 4$$

31. E : Oui
32. Es : Non
33. P : Est ce qu'il y a des basculements de signe ? Il y a un risque de basculement de signe ou quelque chose comme ça ? Si c'était 1,3, par exemple. 9 fois 1,3, ça fait combien ça ? (…) 9 fois 1,3, un peu près ? Ca fait un peu près moins 11, enfin, moins 9 fois 1,3, ça fait moins 11, moins 12, donc le dessus va rester positif. Ca fait pas un changement de signe, ah ? Et en dessous, est-ce que ça change fondamentalement les choses ?
34. Es : Non
35. P : Non plus, d'accord ? Donc, bon, bien sûr, on commet une erreur si on remplace 1,257 par 1 mais ça change pas fondamentalement l'ordre de grandeur. Voilà, les genres d'essais que vous pouvez tenter de, que vous pouvez faire. Vous aviez une feuille de brouillon pour vérifier déjà si le calcul que vous proposiez était bon. Vous pouvez peut-être estimer l'ordre de grandeur sur le papier et ensuite, vérifier, comparer aux calculs que vous faisiez sur la calculatrice. Et pour z, alors, ça donnera combien ? (…) Donc si on retient une valeur proche de 4 pour y, pour z ça ferait ? Caroline ? En fin, le calcul de z est ici (il point sur l'expression écrite au tableau).
36. E : //Moins 1, moins 1//
37. P : Ca fera, ça fera peut-être, oui, aux alentours de moins 3. Ah, ce qui veut dire…On est assez proche de zéro là. Ce qui veut dire que si on a sous-estimé la valeur de y ici, on obtient une valeur négative alors que devrait peut-être obtenir une valeur ? Positive. Donc il y a peut-être une incertitude sur le signe, d'accord ? Mais sur l'ordre de grandeur, on devrait trouver quelque chose qui est assez proche de zéro. Et puis, on peut se poser la question pour savoir ensuite, est-ce que c'est négatif ou non ?

Au tableau :

$z \approx$

$z \approx 0$ négatif ?

38. P : Bon. Voilà, donc, deux premières remarques sur euh, sur les, l'usage de la calculatrice. Et puis, enfin, un petit mot sur l'usage de mémoire. Alors, je sais plus exactement ce qu'on a dit le vendredi. Mais vous allez me, me redire. Donc, sur les calculatrices qu'on vous a proposé, il a quoi comme mémoires ?
39. Es : A, B, C
40. P : Il y avait A, B, C
41. E : Ans
42. P : Et Ans. Ca se comporte fondamentalement pareil ces trois, enfin c'est deux types de mémoires ?
43. Es : Ouais.
44. E : Il y en a une…
45. P : Pierre
46. Pierre : Enfin, il y en une qui s'efface petit à petit et//
47. P : Donc, Ans la mise à jour se fait comment ?
48. Es://
49. P : Mise à jour ? Eve, à quel moment pour la mémoire Ans ?
50. Eve: Mettre au résultat
51. P : C'est-à-dire, quelle touche ?
52. Eve : Égal
53. P : (Il écrit au tableau) Ah, mise à jour à chaque fois qu'on appuie sur « égal », d'accord ? Donc, finalement la mémoire Ans, il faut un petit peu s'en méfier parce que, elle change, au fur et à mesure. Euh, la mémoire A, Faustin, comment est-ce qu'on fait pour la mettre à jour, pour mettre une nouvelle valeur dedans ?

Au tableau :

A, B, C
Ans → mise à jour à chaque fois qu'on appuie sur =

54. Faustin : On tape la valeur.
55. P : Ouais, par exemple, 25
56. Faustin : Et Sto
57. P : Sto
58. Faustin : Et puis, après sur A
59. P : A, d'accord ? Et si on veut utiliser, la valeur de A ?
60. E (un autre) : On appuie sur A
61. P : On appuie sur A et on fait les calculs avec. Alors, il y a un autre moyen que vous avez pas du tout utilisé, c'est la touche Rcl. Vous faite A Rcl, Rcl qui signifie recall. Alors, qu'est ce qui se passe quand vous appuyez sur A Rcl, et bien, c'est la valeur qui est mémorisée apparaît à la place da A. En même temps, ça vous donne un contrôle sur la valeur que vous utilisez. (Une personne de l'établissement vient contrôler l'effectif de la calsee) Ah, il faut dire combien vous êtes. Bonjour, il y a une absente. (À la classe) Voilà des questions là-dessus. C'est bon Léo, des questions ?

Au tableau :

25 Sto A
A …
A Rcl (Recall)

62. Léo : Non
63. P : T'est sûr ?
64. Léo : Ouis ;
65. P : Bien, et on va corriger des exercices que vous aviez à faire.

II. Protocoles de binômes observés

II.1. Binôme la classe 1er V : d'Alpro « machine arithmétique » à Alpro « machine à mémoire Ans » (chapitre D1, 3e partie)

1. Ppr (en distribuant les feuilles de brouillon) : Je vais vous expliquer après avoir distribué les feuilles…Comme vous pouvez le lire, dans cet exercice, vous avez trois phases à faire. « sur la fiche navette, écrivez un message le plus court ». Je vais vous donner les brouillons tout à l'heure et la fiche navette maintenant. Un message qui indique la façon de calculer les expressions et le groupe récepteur n'aura que Alpro, ce groupe ne peut utiliser ni stylo ni papier. Dans le message n'indiquez que les touches de la calculatrice. En suivant ces instructions, ce groupe récepteur doit obtenir la valeur numérique de z à l'écran. Je vous distribuerai les fiches navettes. Alors, vous allez écrire votre message ici et après 10 minutes, vous l'échangez avec le groupe à côté…Donc cette feuille pour la 1ère phase, après 10 minutes, vous devriez terminer votre message afin de pouvoir calculer l'expression z. Pour certain, z est égale à $y^2 - 5y + 7$ avec y qui est une expression fractionnaire de x où avec x = 1.254. Donc vous écrivez exactement ce que vous faites pour que le groupe récepteur dans la 2e phase quand il suit vos instructions, il peut trouver la valeur de l'expression comme vous voulez. Donc, vous avez 10 minutes pour écrire le message. Maintenant vous l'écrivez et puis on l'échangera.
2. Es : Qu'est-ce que c'est un message, monsieur ?
3. Ppu : Un message c'est-à-dire vous écrivez votre façon de calcul.
4. E : Par exemple, lorsque j'enfonce le nombre 3, je dois écrire le nombre 3 ?
5. Ppu : Oui, c'est ça. Appuyer le nombre 3, on écrit le nombre 3. Comme tout à l'heure vous avez compté. Vous appuyez, par exemple, 2, non (il écrit au tableau) 3.141, vous écrivez la façon de l'écrire dans votre feuille pour que le groupe récepteur, lorsqu'il reçoit votre message, il peut suivre ceci.
6. E (à ses camarades) : C'est-à-dire on écrit ce qu'on fait.
7. Ppu : N'oubliez pas que le groupe récepteur n'a rien autre que Alpro et votre message. Je vous distribuerai les brouillons. Chaque groupe a une feuille de brouillon.

Les réponses d' E1, E2 à l'exercice 1 :

E1 (en travaillant sur Fx 570 MS)
Question 1 : 23.872762
Question 2 : 262.9403378
Nombre d'appuis de touche : 54
E2 (en travaillant sur Alpro)
Question 1 : 23.872762
Question 2 : 296.888424
Nombre d'appuis de touche : 80

8. Ppr : (Au binôme E1, E2) Je vais vous enregistrer ce groupe donc, s'il vous plaît, parlez fort.
9. E1 : Mon dieu !
10. Ppu : Vous écrivez aussi votre nom et prénom sur le brouillon.
11. E1 : Ah, bon ? Attendez, ou on met x en facteur ? Alors le calcul est x multiplié par $4x^2 - x$, non $4x^2 - 9$…Ou on prend quand même 1.257 comme Ans ?
12. E2 : Non, voyons, 1.257. Écris le, 1.257 égale AC.
13. E1 : Egale et AC ? (Elle tape sur Alpro)
14. E2 : Oui, Egale et ensuite AC pour que ça disparaisse. Et puis encore AC, oui ça va comme ça.
15. E1 : Puis il faut ON ?
16. E2 : Non, AC. Ca va, puis 4 multiplié par Ans, multiplié par Ans, par Ans.
17. E1 (elle tape sur Alpro) : Je prends un brouillon.

18. E2 : 4 multiplié par Ans, multiplié par Ans.
19. E1 : Donne moi un brouillon. 4 multiplié par Ans, multiplié par Ans, multiplié par Ans et puis ? Moins 9 multiplié par Ans.
20. E2 : Exacte.
21. E1 : Plus 21 et divisé par, divisé par, divisé par…31,2 multiplié par x ah, non, Ans multiplié par Ans, d'accord ? Moins 26 virgule 7184.
22. Ppu : Les groupes n'ont pas encore droit de se discuter entre eux. Vous discutez seulement au sein du groupe.
23. E1 : Puis égale et ceci c'est AC/ON, n'est ce pas ? On doit faire AC/ON puis Ans pour continuer. N'est-ce pas ?
24. Ppu : Vous ne pouvez discuter qu'avec vos camarades dans votre groupe. Et non pas avec d'autres. Deux groupes se trouvant l'un à côté de l'autre ne doit pas parler entre eux. L'un dicte et l'autre tape sur Alpro.
25. E2 : Voyons…
26. E1 : Ouis, puis c'est Ans. Car parce que lorsqu'on a trouvé le résultat.
27. E2 : Mais c'est la division, que doit-on faire ?
28. E1 : Egale, ceci est AC/ON n'est ce pas ? Et on continue de faire Ans pour ceci. Oui, AC/ON, n'est-ce pas ?
29. E2 : Mais ici, il y a une division et nous on a pas de parenthèses, que doit-on fait ? C'est difficile.
30. E1 : Ah, oui…Mais et on ne doit pas écrire sur le papier pour enregistrer le résultat ah ? Ah, oui, c'est possible, j'ai trouvé un moyen.
31. E2 : Comment ?
32. E1 : On va faire comme suivant. Ce x égale Ans, n'est ce pas ?
33. E2 : Mais n'oublie pas que cette valeur est seulement celle d'y. Alors que z c'est y^2 moins 7 y…
34. E1 : Ouis, je sais, mais doucement. Ceci est égal à Ans, n'est ce pas ? Et on peut mémorise le résultat par AC/ON, c'est d'accord ? AC/ON et on mémorise ce résultat. Et maintenant c'est…
35. E2 : On va factoriser z, notre z est égal $y^2 - 7y + 9$ c'est…
36. E1 : Mais on a seulement 10 minutes. Ca set à quoi de diviser par ce bloc ?
37. E2 : 3 et 34…1936. Oh, là là, il n'y a pas de solution.
38. E1 : Alors, si on fait Ans ceci, on considère que ceci est Ans. Donc c'est fait. Mais alors le dessous ? Mais qu'est ce qu'on fait avec le dessous ?
39. E2 : C'est impossible de le mettre en produit.
40. E1 : Oui, quoi, maintenant on cherche à trouver la solution avec ce bloc ? Car si on le mémorise, on sait pas comment on calcule le dessous ?
41. E2 : Ouais. On calcule celui-ci, n'est pas difficile, celui-ci aussi pas difficile, mais le problème est comment diviser celui-ci par celui-ci.
42. E1 : Ouais.
43. Ppu : Attention, il faut que vous notiez votre nom et prénom sur les feuilles.
44. E2 : Tiens, par exemple, si on fait Ans encore une fois, ce sera celui-ci lui-même et non pas celui-ci.
45. E1 : Oui et on peut diviser par ce bloc là mais on ne peut pas diviser par le produit de ces deux là. Le Ans-là, est-ce qu'il garder ça ?
46. E2 : Ah, oui, oui, voici, on va calculer celui-là d'abord
47. E1 : Ouais.
48. E2 : Puis le résultat, on fait Ans. Et puis on prend celui-là et on le divise par Ans.
49. E1 : Oui, c'est vrai, mais si on prend celui-là puis diviser par Ans, on ne peut pas faire Ans avec celui-là. D'accord, c'est-à-dire on doit laisser celui-là, 1.257 alors on doit utiliser plusieurs touches.
50. E2 : Non, non, on va quand même utiliser Ans à la place de x.
51. E1 : Oui, c'est ça et on calcule celui-ci d'abord.
52. E2 : Ouais et le dessous, on laisse, on doit pas toucher…on doit garder 27.75
53. E1 : Ah, on multiplie par 1 sur. Mais 1 sur ceci, non, ce n'est pas possible. Ou sinon, on calcule celui-ci d'abord ? Non, ce n'est pas non plus possible. Si on calcule d'abord celui-ci puis diviser par Ans alors c'est la division par elle-même. On a pas de parenthèse, alors, comment va-t-on faire ?
54. E2 : Regarde, le dessous c'est très super - facile à calculer. Il faut remplacer x en une seule fois. Voyons ça.
55. E1 : Ouais.
56. E2 : Ah, oui, c'est ça, c'est ça…C'est facile.
57. E1 : Mais où on mémorise le résultat ?
58. E2 : Oui, c'est ça, 1.2 égale…
59. E1 : Oui, ou on calcule quand même celui-ci. Oui, on calcule celui-ci. Oui on calcule celui-ci d'abord.
60. E2 : Et on calcule ça après. Mais on peut pas.
61. E1 : Si, on le calcule maintenant. On vient de le faire, c'est ça ? Allez, calcule le car maintenant il suffit de lui demander de diviser une seule…valeur du dessous. Alors, fais le.
62. E2 : Non, ce n'est pas possible. C'est facile de le comprendre entre nous, mais ce n'est pas possible.
63. E1 (elle relit l'énoncé) : « Ne pas utiliser du papier et stylo mais seulement la calculatrice Alpro»…
64. E2 : C'est-à-dire on ne peut pas utiliser des outils extérieurs pour le calcul.
65. E1 : Ouais (elle demande au groupe à côté) T'as déjà calculé le dessous ? C'est combien ?
66. E (du groupe à côté) : 12.5
67. E2 : 12.5 ! Oh, ce nombre est beau, ce résultat !
68. E1 : Ok, maintenant, on prend celui-là divisé par 12,5
69. E2 : Non, on ne peut pas faire comme ça. D'où on trouve le nombre 12.5.
70. E1 : Oui, mais on le sait. Pourquoi pas ?
71. E2 : Oui, c'est OK, ça fait rien.
72. E1 : N'est ce pas. En tout cas, il le saura.
73. E2 (Il lit les instructions puis E1 les tape sur AlPpu) : Multiplier par Ans, Multiplier par Ans,

Multiplier par Ans puis moins 9 Ans plus 21. Pourquoi ils peuvent appuyer comme ça ? Egale.
74. E1 : Egale et divise par ?
75. E2 : Oui, tu peux l'essayer. Egale. Puis divise par 12,5. Non, non, il faut un égale.
76. Ppu : Vous ne discutez pas entre des groupes. Vous commencez à écrire le message. Il vous reste seulement quelques minutes pour cette phase. Je vous répète que vous commencez à écrire le message dans votre feuille.
77. E1 : Egale et puis divise par 12.5 ? N'est-ce pas.
78. E2 : Ouais, égale et c'est fini.
79. Ppu : Et puis je ramasserai les feuilles dans 2 minutes.
80. E1 : Et on calcule y, n'est ce pas ?
81. E2 : x au carré.
82. E1 : Quoi au carré ? Maintenant on « Ans » celui-ci, on « Ans » celui-ci.
83. E2 : Egale. Ouais.
84. E1 : AC/ON, n'est ce pas ?
85. E2 : Ouais, c'est ça, c'est ça. Maintenant on transforme Ans en y. Ans au carré.
86. E1 : Ans au carré et quoi ?
87. E2 : Moins 7 Ans, plus 9 et égale puis c'est fini.
88. Ppu : Maintenant, vous allez exécuter exactement ce que vous avez écrit pour voir si c'est correct ?
89. E1 : On fait avec ceci pour voir si c'est bien ?
90. E2 : Maintenant, on fait des essais. On fait en suivant exactement ce qu'on écrit pour voir si c'est correct. Tu utilises cette calculatrice-ci pour voir si c'est correct ?
91. E1 : Non, avec celle-là on ne peut pas. Il faut essayer avec l'autre.
92. E2 : Avec celle-là pour voir si notre résultat est correct ?
93. Ppr : Est-ce que vous avez marqué vos noms ?
94. E1 : Attendez, laissez nous du temps pour le faire.
95. Ppu : Et pas sur la 2e partie d'échange.
96. E2 : Maintenant tu suis exactement ce que nous avons écrit pour voir…AC/ON. Voyons ça, il semble que c'est correct.
97. E1 : Il me paraît qu'il n'y a pas le signe « carré » ? N'est-ce pas ?
98. E2 : Ans multiplié par Ans. C'est OK.
99. E1 : Toi aussi, tu essaies de calculer. Essaie avec la calculatrice de poche.
100. E2 : 1 virgule…
101. E1 : Virgule quoi ? 57
102. E2 : 1.257, égale et AC/ON.
103. E1 : 4 multiplié par Ans…
104. E2 : 4 Multiplié par Ans, multiplié par Ans, multiplié par Ans, égale.
105. E1 : Le signe moins. Moins 9.
106. E2 : Moins 9 multiplié par Ans.
107. E1 : Plus 21.
108. Ppu : Il vous reste 2 minutes. Dépêchez vous. N'oubliez pas votre nom. Vous l'échangez quand je vous le dites.
109. E2 : Je suis sûr que ce ne peut pas être faux notre calcul.
110. E1 : Egale. On peut diviser tout de suite ?
111. E2 : Oui, bien sûr. Il n'y a rien qui est faux.
112. E1 : Divise par 12.5 puis égale. AC/ON.
113. E2 : C'est bien fait. On a gagné !
114. E1 : Ans multiplié par Ans.
115. E2 : Alors, attends, on essaie pour voir si c'est OK. Oui, oui, c'est bon. Ans multiplié par Ans, puis moins 7 Ans. Plus n'est ce pas ?
116. E1 : Plus 9. Tu vois. C'est pas faux. On l'espère en tout cas. On peut diviser par ce résultat ? Faire comme ça ? Personne ne l'interdit, n'est-ce pas ?
117. E2 : Si l'on ne suit pas cette méthode alors on …
118. E1 : N'est-ce pas. Ici on peut utiliser du papier, de stylo et la calculatrice. Mais est-ce qu'on peut diviser en utilisant un résultat ? Alors, essayons de trouver un autre moyen.

Le message de ce binôme :

= AC/ON

$4 \times Ans \times Ans \times Ans - 9 \times Ans + 21 = \div 12.5 =$ AC/ON

$Ans \times Ans - 7 \times Ans + 9 =$

119. Ppu : Est ce que vous avez tout terminé ?
120. E1 : Oui, c'est déjà terminé.
121. Ppu : Maintenant vous vous arrêtez puis vous échangez vos feuilles. Ecoutez moi, s'il vous plaît. Deux groupes s'échangent et vous suivez les instructions de vos camarades, puis vous notez dans la phase 2 sur cette feuille (il a pris une feuille navette et l'a montré devant toute la classe) ce que vous aurez obtenu après l'exécution du message de l'autre groupe. Ce que vous avez sur l'écran d'Alpro. Alors vous le faites maintenant et vous aurez 5 minutes pour le faire. (Il contrôle l'échange entre les groupes) //

Le message reçu :

= AC/ON

$2 \times Ans \times Ans \times Ans + 15 \times Ans - 5 =$ (1) AC/ON

$3.12 \times 1.254 - 27.7534 =$ (2) AC/ON

$\frac{(1)}{Ans} =$ (3) AC/ON

$z = Ans \times Ans - 5 \times Ans + 7$

122. E1 : On commence à l'exécuter, ah ? On commence. 1 virgule 2…
123. E2 : Le dernier résultat, c'est combien ?
124. E1 : Alors, 1.257, égale AC/ON.
125. Ppu : Vous faites attention, ah. Par exemple, si c'est le groupe A qui reçoit le message du groupe B alors ceux dans le groupe A notent leur nom dans la cage qui est réservée à cet effet.
126. E1 (elle dicte à E2 qui tape sur Alpro) : Oui. Laisse moi noter les résultats. C'est bon, c'est bon. On commence, 2 multiplié par Ans, multiplié par Ans, multiplié par Ans, plus 15 multiplié par Ans. Pourquoi c'est 15 multiplié par Ans ? Moins 5. Egale. AC/ON.

127. E2 : Oui, c'est fini, AC/ON et puis ?
128. E1 : Pourquoi ils peuvent mettrent Ans plusieurs fois comme ça ? Ce Ans ne peut pas mémoriser le résultat, n'est-ce pas ?
129. Ppu : Vous devez maquer votre nom dans la feuille reçue dans la partie phase 2 et vous avez 5 minutes pour terminer cette partie.
130. E1 : On continue, 3.12, oui, 3.12. On ne peut pas utiliser du papier et du stylo, comprends ? 3.12. Multiplié par 3.1 non multiplié par 1.254.
131. E2 : 1.254 ou 1.257 ?
132. E1 : 1.254. Non c'est 1.257. Non, je me suis trompée depuis tout à l'heure. On recommence ?
133. E2 : Non, tant pis pour toi.
134. E1 : Oui, on commence alors. Moins 27.7534 et égale. AC/ON. Maintenant ce groupe prend cet Ans et 1 sur ça. Mais on ne peut pas prendre ceci pour diviser par Ans.
135. Ppr : Avez-vous déjà marqué votre nom ?
136. E1 : Monsieur ? Si c'est par exemple comme ça, comment on…
137. Ppr : Vous suivez exactement les instructions de l'autre groupe. Si vous n'y arrivez pas, notez ce que vous obtiendrez sur l'écran d'Alpro.
138. E1 : Mais si on ne peut pas ? S'il y a une faute ?
139. Ppu : S'il y a une annonce ou un résultat, vous le notez ici. Notez ce que vous avez sur l'écran. Vous faites suivant les instructions de vos camarades. Par exemple, l'étape 1 puis l'étape 2 qui ne marche pas, et Alpro vous sort une annonce et vous le notez.
140. E1 : Oui et part exemple si Alpro dit que c'est une faute, on doit noter ?
141. Ppr : Oui, vous le notez et puis n'oubliez pas de marquer votre nom.
142. E1 : Tu as fini de taper AC/ON ?
143. E2 : (du groupe émetteur) Nous avons oublié de noter le résultat déjà trouvé.
144. E1 : Diviser par Ans. Oui, diviser par Ans et puis AC/ON.
145. E2 (lit l'annonce sur Alpro) : Expression non valide.
146. E1 : Mais c'est pas très « erreur », on recommence. Tu appuies 1.254.
147. E2 (il dit en français) : On ne peut pas,
148. E1 : Alors, 1.254, AC/ON appuie AC/ON. Dès le début. 254, AC/ON. Ok, 2 multiplié par Ans, 2 multiplié par Ans, 2 multiplié par Ans plus 15 multiplié par Ans moins 5, égale AC/ON. Ok, c'est bon. 3.12 multiplié par 1.254 moins 27.7534, égale, AC/ON. Ouis…Euh…Alors maintenant comment fait-on ?
149. E2 : Le résultat que tout à l'heure on a trouvé est – 1.26.
150. E1 : Oui, c'est OK. Il y a pas d'erreur. T'as trouvé ?
151. E2 : 1 virgule…
152. E1 : C'est quoi, c'est le résultat de la division ?
153. E2 : Oui, c'est ça. Leur Ans c'est justement ça et (1) c'est ça. On les a déjà calculés, mais on a oublié. Il faut recommencer de nouveau alors.
154. E1 (elle répète) : On recommence quoi ? Non, Il est impossible de calculer. Tais toi, on continue alors Ans.//
155. E2 : 1 virgule.
156. E1. Alors, on continue. Alors Ans multiplié par Ans. Non, égale, diviser par Ans, oui diviser par Ans.
157. E2 : Mais si on divise par Ans, ça sera 1 ?
158. E1 : Mais ils ont fait comme ça, tant pis pour eux.
159. E2 : Ah, oui. Ils ont trouvé ce résultat, puis ils ont fait AC/ON.
160. E1 : Puis ils prennent le résultat auparavant.
161. E2 : Et ils ont pris ce résultat et ils ont divisé tout de suite par Ans.
162. E1 : Ouais, égale.
163. E2 (au groupe émetteur) : /Il faut que vous n'écriviez pas (1) pour qu'on sache le résultat./
164. E1 : Recalcule quoi ? Non, ce n'est pas ça. Tu écris, Ans… divise par Ans.
165. E2 : Si on divise par Ans c'est 1 ?
166. E1 : Oui.
167. E2 : Ils calculent ce résultat, puis ils divisent par Ans…
168. Ppu : Il vous reste 2 minutes pour terminer ça et puis je vais continuer la 3 phase.
169. E1 (elle lit l'écran) : Erreur syntaxe. Expression non valide.
170. E2 : Tu notes seulement erreur.
171. E1 : Oui, note le vite. C'est fini…
172. E2 : Il y a combien de question au total ?
173. L'écran obtenu après avoir exécuté le message :

Erreur Synt. Expression non Valide

174. Ppu : Maintenant, toute la classe, vous arrêtez. Alors, attention, s'il vous plaît, écoutez moi bien. Maintenant, vous redonnez le message que vous avez reçu au groupe émetteur et vous recevez ancien message. Alors, dans votre feuille, vous pouvez voir que sur le message reçu, vous pouvez constater que vos camarades ont écrit, soit le résultat, soit l'erreur. Donc en se basant sur ça, à votre avis, vous pouvez corriger ou modifier le message et puis noter le nouveau message en fonction des corrections de vos camarades, dans la phase 3 pour calculer la valeur de l'expression, la valeur de z et vous aurez 10 minutes. En fonction des remarques de vos camarades, vous pouvez aussi recorriger votre message. Par exemple, si vous voulez que ça devrait être mieux fait vous le corriger sinon, si c'est correct vous ne modifiez rien.
L'écran obtenu par l'autre groupe après l'exécution du message de ce binôme :

Ans × Ans – 7 × Ans + 9
1.115931226

175. E1 : Il faut penser un autre moyen pour diviser par cette valeur sans qu'on doive noter ce résultat.
176. E2 : Ce que nous avons fait c'est aussi une méthode.
177. E1 : Oui mais est-ce qu'il y a une autre ?

178. Ppr : Si vous jugez que c'est correct vous pouvez modifier rien //
179. E1 : Ah, je l'ai trouvé. Il y a un moyen. On a déjà trouvé le résultat du haut, n'est ce pas ?
180. E2 : Mais où on va enregistrer le résultat ?
181. Por : Si vous avez trouvé correct ce qu'a noté l'autre groupe, vous le laissez. Si vous pensez que ce n'est pas comme ce que vous vouliez faire, vous le corrigerez et vous marquez dans la 3e cage.
182. E2 : Il semble que la méthode où on doit noter le résultat est plus juste.
183. E1 : Tais-toi, je te dis que…
184. E2 : Et ce n'est pas comme ils ont calculé le résultat pour l'autre comme ils ont fait. Nous on a divisé directement par 12.5.
185. E1 : Et alors j'ai trouvé une méthode. 1.257 AC/ON et…
186. E2 (en français) : Comment, comment ? 1.257, égale puis AC/ON, et puis après, 4 multiplié par Ans, 4 multiplié par Ans, multiplié par Ans et moins 9 multiplié par Ans plus 21 et égale. Puis AC/ON. C'est AC/ON qui a donc gardé ce résultat. Et on obtien, Ans divisé par…Est ce que c'est cette somme divisée par ce produit ? N'est-ce pas ?
187. E2 : Ouais, ouais.
188. E1 :Donc est ce que c'est égal à la somme de chaque terme divisé par ce produit ?
189. E2 : Non, tu rêves ou quoi ? Ceci est divisé par ceci plus ceci divisé par ceci.
190. E1 (au brouillon) : Non, c'est ce que A divisé par A moins B est égale à A, c'est-à-dire égale à A multiplié par 1 sur A moins 1 sur B ?
E1 écrit au brouillon :

$A : (A - B)$ $\quad A\left(\frac{1}{A} - \frac{1}{B}\right)$

191. E2 : Non, non, ce n'est pas égale…7 minutes de décalage…
192. Ppu : Est ce que toute la classe a déjà fini ? Maintenant, je vais ramasser et je vais demander à un groupe de venir ici pour exécuter son message devant les autres sur cet ordinateur et les autres observent.
193. Ppr : Vous avez rien à ajouter ?
194. E1 : //le résultat.
195. Ppr : Votre résultat est-il correct ?
196. E1 : Je ne sais pas…Tu verras si le résultat est correct ? (au groupe à côté) comment vous avez fait ? Pas comme ça ? Alors comment faire ?
197. E2 : Alors ainsi nous, on a pas compris. Il faut avoir quelqu'un pour l'expliquer.
198. E1 : Ah, bon. Comment ils font alors ?
199. E2 : Par exemple, 20 ils ont fait Ans et puis 4, ils ont fait, non, non. Ils posent.
200. E1 : Attends, 4…Non, ce n'est pas ça. 20. AC/ON, n'est ce pas Voyons 12.5 égale A, et…
201. Ppu : Allez le groupe de N et D.
202. E1 : Oh, tiens, c'est possible. Oh, là, ah, je l'ai trouvé, je l'ai vu.
203. Ppu : Regardez, toute la classe, s'il vous plaît. Toute la classe, vous observez ce que fait ce binôme.
204. E1 : On n'a qu'à ajouter : 12.5 égale A. Just. // et 12.5 égale A.
205. Ppu : Alors c'est le groupe de N. et D.
206. E1 : Oh, tu sais ça y est, j'ai trouvé.
207. Ppu : S'il vous plaît. Vous observez ce que fait ce groupe. Et vous (aux élèves du groupe), vous exécutez votre message. (La classe observe ce que ce groupe fait)
Le dernier message du binôme N et D :
1.257 = Ans × Ans × Ans × 4 – 9 × Ans + 21 = 17.63148637
1.257 × 31.2 – 26.7184 = ~~13.0616~~ ~~12.5~~ 12.5
17.63148637 : ~~13.0616~~ 12.5 = 1.410518916
Ans × Ans - 7 × Ans + 9 = 1.115931227

208. Ppu : Ce groupe doit calculer avec x = 1.254.
209. E (du groupe) : Non, c'est 1.257.
210. Ppu : Oui, 1.257. Oui, vous continuez et les autres, vous y faites attention. Ils doivent calculer…Ces deux expressions sont assez analogues…
211. E1 : Alors, ainsi on ne doit que calculer ça et on pose 12.5 égale à A.
212. Ppu : Voilà, attendez, arrêtez par là. Donc ce groupe a calculé par la méthode suivante : ils ont tapé 1.257 puis égale et Ans. Alors, le résultat obtenu au tableau est le numérateur de l'expression y. N'est-ce pas ? Alors vous pouvez continuer. C'est le numérateur d'y. Oui, tu continues.
213. E1(au binôme au tableau) : Tu ne peux pas écrire.
214. Ppu : Tu continues en suivant les instructions dans ton message. C'est le numérateur d'y, vous y êtes ?
215. E1 : Je ne sais pas ces touches.
216. E2 : Moi, non plus.
217. E1 : Mon dieu…
218. Ppu : Silence s'il vous plaît. Et puis ils ont mis y dans A…S'il vous plaît, vous suivez ce qu'ils font, ah…(Ce groupe se plante !) Vous voyez, même vous, vous arrivez pas à exécuter votre message. Plus concrètement, vous avez écrit des choses implicites dans votre message donc lorsque vous devez ré-exécuter, vous vous y perdez un peu. Bon, regagnez vous votre place. Alors, je vais inviter un autre groupe.
219. E2 : Ils ont pas appuyé Sto, Stec, n'est ce pas ? Par exemple, si on veut mettre 12 dans A, comment fait-on ? Si on veut avoir 12 égale à A, comment fait-on ?
220. E1 : // et puis AC/ON, d'accord. Puis 52 divise par A, c'est-à-dire par 12. Non, il est impossible de diviser. Ce n'est pas vrai n'est ce pas ?
221. Ppu : Je demande le groupe d'H et T de venir au tableau.
Le dernier message de ce groupe :
= A
= Ans
Shift Sto A AC/ON A × A × A × 2 + 15 × A – 5
= ~~Ans~~

Shift Sto B AC/ON 32.1 × A – 27.7534 = B : Ans =

Ans × Ans – 5 × Ans + 7 = 1.1915731330

222. Ppr : Avec quelle valeur dois tu calculer z ? Ton x est égale à comme bien ?
223. E (du binôme) : 1.254
224. Ppu : Alors ils doivent calculer l'expression avec x = 1.254. Ces deux expressions sont semblables, mais les valeurs de x sont un peu différentes.
225. E2 : C'est pas possible. Car 13 multiplié par 12 pas égale à 52.
226. E1 : 12…
227. Ppu : Vous les observez bien, s'il vous plaît. Vous observez, s'il vous plait. 1.254, d'abord ils font, ils appuient Shift Sto A, c'est-à-dire stocker cette valeur de x dans A. Puis ils calculent le numérateur y qui est égale à B, puis stockent dans B…
228. E1 : Ca vient d'où 15 ?…C'est 26,7184 et non pas 27,7534 ?
229. Ppu (après qu'ils finissent d'exécuter le message) : Bon, est-ce que vous comprenez leur méthode ? Alors 1.254, (il écrit en même temps au tableau), ils mémorisent 1.254 dans, en utilisant les touches Shift, Sto et A. Alors, comme ça, on a la valeur 1.254. Et puis, pour calculer l'expression y, non, le numérateur d'y, ils prennent A × A × A × 2 + 15 × A – 5 =, puis ils l'enregistrent dans B (il efface le signe = et y écrit la flèche et B). Puis ils calculent le dénominateur d'y. Ils prennent 32.1 multiplié par A, c'est-à-dire x et moins 27.7534, ils…Ici vous l'enregistrez où ou vous le laissez ?
230. E (du binôme au tableau) : On le laisse.
231. Ppu : Alors oui, le signe égale. Puis vous continuez comment ?
232. E : A divise par cet Ans.
233. Ppu : Ouis A divise par Ans.
234. Es : Non, c'est B divise par Ans.
235. E1 : C'est B divise par Ans.
236. Ppu : C'est quoi alors ?
237. E : C'est Ans divise par B.
238. Es : Non, B divise par Ans
239. Ppu : Oui c'est la valeur enregistrée dans B qui est le numérateur. Ici c'est B divise par Ans.
Au tableau :

→ A

Numérateur d'y : A × A × A × 2 + 15 × A – 5 → B

Dénominateur d'y : 32.1 × A – 27.7534 =

~~A ÷ Ans~~

B ÷ Ans

240. Ppu : Qui a une autre méthode ?…Non. Alors ici c'est calculer…Pourquoi doit-on mémoriser l'expression, pourquoi doit-on mémoriser x dans A ? La valeur 1.254 dans A ?
241. E (du binôme) : Pour ne pas devoir re-écrire la valeur 1.254.
242. Ppu : Et si on l'avait pas fait. Est-ce qu'on pourrait le faire ?
243. E : Oui, on fait avec Ans.
244. Ppu : Mais dans ce cas, comment on calcule le dénominateur ?
245. Es : On doit noter le résultat du numérateur.
246. Ppu : Donc on doit noter le numérateur, n'est-ce pas ?
247. Es : Ouis.
248. Ppu : Mais vous le savez, le groupe récepteur, ils peuvent pas, on ne peut pas écrire. Alors, on doit enregistrer x dans A. Vous faites une pose de 5 minutes et on l'attaquera la 2e partie. N'oubliez pas que vous n'avez que 5 minutes.
249. E1 :J'ai déjà compris mieux plein de chose. Par exemple, la façon de mémoriser. On sait comment enregistrer un nombre alors. On appuie Shift, Sto et on enregistre le résultat.
250. E2 : Alors j'essaie de calculer ceci. C'est 12 et A, c'est ça ?
251. E1 : Non, ce n'est pas 12 A. C'est 12 Shift, Sto A. Sto A.
252. E2 : Oui, OK. Donc dans A, il y a un nombre. 36. Maintenant.
253. E1 : 36 Sto B. Oui, c'est fait.
254. E2 : B ÷ A. Egale. 3. Oui c'est bon.
255. E1 : Tu vois, c'est OK. J'ai compris. Elle tourne encore la cassette (…) Oh, regarde, tu vois, par exemple 15, égale, A AC/ON, A. Non ça marche pas.
256. E2 : Il manque AC/ON. Comme tout à l'heure, 15 égale à A, n'est ce pas ?
257. E2 : Essaie avec A divise par 3 pour voir.
258. E1 : Oui, A divise par 3 A ÷ 3.
259. E2 : Egale 4. 15 ÷ 3 égale à 4 ?
260. E1 : Non, je recommence, tout à l'heure, je me suis trompée. 15, égale, A ÷ 3. Non ça marche pas. 15 divisé par 3 égale 4 ?
261. E (du binôme à côté) Stocké, stocké, ça veut dire quoi ?
262. E1 : C'est enregistré. Voyons ça, (elle lit) « Stocker une valeur dans une mémoire variable, ça».
263. E2 : Alors, la prochaine fois, on n'a qu'à suivre les instructions et ça avant de faire les calculs.
264. E1 : Oui, exacte, c'est ça. (Elle lit ce qui est apparu sur l'écran lorsqu'elle déplace le curseur près des touches) Ajouter une valeur affichée sur l'écran dans…Oui, attends, attends. Ajouter une valeur dans…
265. E2 : Ah, on peut utiliser Min, ça marche aussi, c'est pareil pour la touche Min. Ca marche aussi.
266. E1 : Pourquoi Min.
267. E2 : Regarde.
268. Ppu : Attention, s'il vous plaît. Dans l'exercice 2, vous avez deux parties.

II.2. Binôme la classe 2nd F2 : de la machine arithmétique à la machine à mémoire A, B, C et Ans (mémoire variables mathématiques) : une évolution suscitée (chapitre D1, 3e partie)

1.Ppu : […] Et puis, ensuite, il faut allumer la calculatrice en appuyant sur la touche « AC/On ». Voilà, vous avez accès à la calculatrice, vous faites ce qu'on vous demande sur la feuille de consignes. D'accord ? Donc dans cette phase là, c'est chacun pour soi…
(Dans cette phase, M et S travaillent individuellement.)
2.E1 : Là, on fait direct ?
3.E2 : Ouais.
4.Ppu : Donc vous vous apercevez, c'est une calculatrice qui est, qui marche pas tout à fait comme les calculatrices normales puisqu'il y a certaines fonctions ne sont pas accessibles, d'accord ? Donc il faut que vous vous débrouilliez avec ce qu'on vous propose, ah ? C'est le but de la manif. Ca va Ophélya là ? (…) Vous marquez bien votre nom en haut de la feuille. Votre nom et votre classe (…)
5.E1 : Y pas de parenthèses, y a rien…Il faut aller dans la séance.
6.Ob. : Faut faire « nouvel exercice » puis « nouvelle question » non « question suivante ».
7.E1 : Donc…Y a pas Pi pour entrer…(il lit ce qui apparaît sur l'écran lors qu'il déplace le curseur sur la touche A) « mettre une valeur dans la mémoire variable A ». Oui, mais je veux bien. // Egale…On tape direct.
8.E2 : Tu vas retaper ça à chaque fois ?
9.E1 : Tu fais comme ça. On doit se débrouiller dans la vie (…)
10. E1 : Comment.
11. Ob. : Tu t'es trompé. 9 fois 8 ça fait pas //
12. Ppu : Vous avez droit à tout ce que vous voulez. Dans deux minutes, je ramasserai les copies (…) Ah, oui, voilà, il y a une chose importante.
13. Es : Oh, non.
14. Ppu : Il y a une chose importante, c'est, que j'ai oublié de le dire, désolé. C'est qu'il va falloir compter le nombre de touches que vous aurez à frapper.
15. Es : Non.
16. Ppu : Mais vous pouvez, vous pouvez l'estimer ça. Même si vous ne faites pas entièrement les calculs.
17. Ppr : Tu comptes un peu près. C'est pas l'unité près ah ? Ce qui compte c'est pas d'avoir l'unité près mais d'avoir l'ordre de grandeur.
18. Ppu : Vous avez vu, dans le 2e exercice. Il faut compter le nombre de touches que vous aurez tapé (…) Donc, vous pensez bien compter le nombre de touches, ah ?
19. E1 : Combien de clicks ça ?
20. Ppr : Pardon ?
21. E1 : Combien de clicks en fait.
22. Ppr : Voilà, combien de clicks (…)
23. E2 : Tu vas retaper ça à chaque fois ?
24. E1 : Mais je suis en train de faire quoi ? On fait 3 millions de touches là (…) Voilà. 297 point 8. T'as trouvé ça ?
25. Ppr : C'est bon ? Le nombre de touches ? Jonathan, c'est bon là ?
26. E1 : Non, non (…)
27. E2 : Tu tapes ça. 3, 14. tapes le. Egale et tu fais Ans, Ans fois voilà. Ans fois Ans. Non. On a 3 fois. Fais égale. Ans fois 8. Parce que c'est ça. Ans fois 8…
28. Ppu : Voilà, alors je vais demander…Déjà, on va mettre peut-être les réponses sur les écrans. Mélodie, tu as fini ? Je peux prendre. Eve. Merci. C'est bon, tout le monde me l'a rendu ? Alors, les bonnes réponses, d'après vos feuilles. Pour les premiers calculs, il y a pas trop de. Vous regardez par là, s'il vous plait.
29. E2 : Tu tapes…
30. Ppu : Jonathan, on arrête là, maintenant.
Leur procédure :

Question a
E1 : 2×3.141×3.141+3.141+1 =
E2 : 2 × 3.141 = ; × 3.141 = ; +3.141+1 = ; + 1= ;
Question b
E1 (les dernières lignes) : 3.141759682 = ; Ans×Ans×Ans = ; Ans x 8 = ; + 6 × 3.141759682
E2 : 3,141759682 = ; Ans × Ans × Ans ; × 8 = ; + 6 × 3,141759682 × 3,141759682 = ; - 3 × 3,141759682 = ; - 1 = ;

Leur réponse :

E1 : Réponse 1 : 23,872762 ; Réponse 2 : ; Nombre de fois :;
E2 : Réponse 1 : 23,872762 ; Réponse 2 : 296,888424 ; Nombre de fois : ≅ 45 ;

31. Ppu : Pour les premiers calculs, il y a un peu près une unanimité. Donc, si je note bien tous les chiffres qui sont donnés (il écrit le résultat au tableau), vous trouvez 23 virgule 8 cent soixante douze…sept cent soixante deux. Pour le 2e, vous trouvé, pour ceux qui ont trouvé le bon résultat, vous trouvez deux cent quatre vingt seize virgule huit cents quatre vingt huit…quatre cent vingt quatre.
Au tableau :
Réponse 1 = 23,872762
Réponse 2 = 296,888424

32. Ppu : Alors je vais demander à Elodie. Donc, vous êtes nombreux à trouver 80 touches, un peu près dans l'ordre de grandeur quatre vingt touches pour calculer ce résultat, le résultat de la question 2.

Je demande d'Elodie qui a trouvé soixante, aux environs 60 touches de passer sur la calculatrice qui est ici et de venir nous montrer ce qu'elle a fait. Vous regardez bien. Oui, je te redonne ta fiche. Et en même temps qu'il y a Nicolas qui va venir contrôler le nombre de touches tapées. Tu viens, Nicolas, tu regardes ce qu'elle fait. Je vais donner ta fiche. Bon, en fait ce n'est pas très, très important mais.
33. E : Non, c'est juste pour voir le calcul.
34. Ppu : Tiens, voilà. Vous regardez bien ce qu'elle a fait. Et ensuite, je vous distribuerai, on passera au 2e exercice. C'est l'exercice 2. Donc, je fais pas de commentaire, pas d'explication. Vous regardez comment elle a fait, ah ? (…) (la classe observe ce que fait Elodie)
35. Ppr : Nicolas, tu comptes en même temps ah ? (…) Voilà, tu arrives au même résultat, t'as trouvé combien ? Soixante quatre. Soixante quatre touches. (Il note au tableau)
Au tableau :
≈ 60 touches
64 touches

36. Ppu : Vous avez vu ce qu'elle a fait ? Alors, il y en a qui a trouvé moins. C'est Stéphane ?
37. E2 : Mais alors là, je me suis trompé.
38. Ppu : Il y a, qu'il a mis quarante touches. Qu'est ce que tu en penses ? Vu la façon dont tu a fait, il y a des chances que tu as mis, t'as passé plus de de touches que soixante, ah ?
39. E2 : Ouais, ouais.
40. Ppu : On est d'accord. Voilà, on va s'en tenir là pour le 1er exercice.
Le procédé d'Elodie :
3,141759682 = ; Ans × Ans × Ans = ; × 8 + 6 × 3,141759682 × 3,141759682 – 3,141759682 – 1 =

41. Ppu : Je vais vous donner un 2e exercice. Mais il est à faire en binôme.
42. E : C'est-à-dire ?
43. Ppu : Alors les binômes, je les donne, il y a, Stéphane et Jonathan. C'est les binômes habituels un peu près ah ? Ensuite, Johanna et Caroline. Donc, vous fermez une des deux calculatrices. Ah, pareil pour Stéphane et Jonathan. Et sur l'autre, vous passez à l'exercice numéro 2. Alors, qu'est ce qu'il y a d'autres ? Il y a Laura et Maude, en semble. Vous vous mettez ensemble, Laura, où elle est, elle est là-bas. Laura et Maude vous mettez en semble, s'il vous plaît. Il y a un trinôme, c'est Caroline, Mélodie et Elodie. D'accord ? Alors vous vous mettez vite ensemble, qu'on perde pas trop de temps. Caroline, Mélodie et Elodie. Venez là. Mélodie, tu viens là. Caroline et Coline. Amandine et Eve, c'est fait. Et Aurélie et Gladys, vous vous mettez ensemble.
44. E2 : Mais non, qu'est-ce c'est que ça ?
45. Ppu : Vous fermez une et vous ne gardez qu'une calculatrice. Vous pensez à passer à l'exercice 2 dans la calculatrice, ah ?
46. E1 : Vas-y.
47. Ppu : Dans l'autre. Vous dites bien, nouvel exercice, numéro 2. //
48. E : Nouvelle question ?
49. Ppr : Oui, voilà, première question. En fait, il y a une seule question dedans.
50. E1 : Vas-y, nouvelle question.
51. E2 : Non, c'est fait.
52. Ppu : Il y a une seule question mais c'est un petit peu plus longue. Donc, je vous passe trois feuilles. Je vous passe une feuille d'énoncé, une fiche navette et une feuille de brouillon. L'énoncé, vous marquez, vous marquez, votre nom dessus. La fiche navette, vous marquez aussi votre nom. Alors, ça se passe comment ? Donc, par exemple, si je prends deux groupes là. Vous avez, les 2 groupes, ils ont des énoncés différents. Ils vont devoir faire un travail et l'échanger ensuite. D'accord ? Mais vous avez un énoncé qui est légèrement différent. Pensez pas que vous aviez le même travail que votre voisin. Je vous distribue les feuilles, vous allez mieux comprendre. Donc, vous marquez bien le nom sur chaque feuille
53. E2 : On marque ça, le nôme du binôme.
54. E1 : Mais à quoi sert la feuille navette ? Je // m'égare.
55. Ppu : Voilà. Tout le monde doit avoir 3 feuilles, normalement. Voilà, vous marquez bien votre nom. Est-ce que tout le monde a la fiche navette ? Il me manque'une. Oui, tout le monde l'a ? Vous l'avez pas ? Pardon ? Ah, voilà, c'est ça. Vous n'avez pas la fiche navette ?
56. E2 : (ils lisent l'énoncé) « Phase 1…Ecrivez un message le plus cour possible à un autre binôme… Suivant vos instructions, la valeur numérique z…»
57. E1 : Uh !!!
58. Ppu : Ah, il y en a un petit…Non c'est pas normal. Donc il m'en faut, il m'en faut deux. Voilà, un petit problème de tirage. Alors, je voudrais insister sur une chose. Vous avez lu dans les instructions dans le message. Je vous laisse lire le message et ensuite, ensuite je vous donne une petite précision. Lisez bien le message.
59. E2 : Ca sert à quoi ? Ils veut nous embrouillez là. Ca sert à quelque chose. Vas-y. On va déjà calculer et après on divise par ça. On va d'abord calculer ces deux lignes et après on va calculer cette ligne.
60. E1 : Ouais, ouais, je sais. 1, 257.
61. E2 : 3 fois. Ans fois Ans fois Ans et fois 4 moins 9 fois Ans plus…
62. E1 : Plus 21. Divisé par, 31 virgule 2…
63. E2 : Ca c'est pas bon par ce que ça va faire ça divisé par 31 virgule 2 puis après ça va faire fois Ans moins 26. Tu vois ce que je veux dire ?
64. E1 : Non.

65. E2 : Ca va juste diviser ça par 31 virgule 2. Et après ça va multiplier par ça.
66. E1 : Et il faudra des parenthèses//
67. E2 : Non, non.
68. E1 : Tu veux qu'on fasse ? Comme tu dis, d'accord ? Vas-y, essaie.
69. E2 : Fois 1 virgule...
70. E1 : Moins.
71. E2 : Moins 26 virgule 7184. Moins ?
72. E1 : Moins 6.
73. E2 : Et après on marque ce que j'ai dit ? On fait 31 virgule fois 1 virgule 257 moins 26 virgule 7184 égale. C'est 12,5. Alors c'est 17 virgule, 37 divisé. Mais on trouve pas pareil là. On trouve pas pareil. Mais on prend celui là.
74. E1 : Mais on peut pas trouvé un nombre négatif en lisant ça.
75. E2 : Donc ça c'est y. On sais combien vaut y.
76. E1 : Ouais.
77. E2 : Et z égale à ?
78. E1 : y au carré moins 7 y plus 9.

Ce qu'ils font jusque là :

1,257 = ; Ans × Ans × Ans × 4 − 9 × Ans + 21 ; 31,2 × 1,257 – 26,7184 = ; 17.63148637 ÷ Ans = ;

79. E2 : Ca fait, et y c'est 05 ?
80. E1 : 1891.
81. E2 : Egale. Ans fois Ans moins
82. E1 : Moins 7 fois Ans plus 9. Egale.
83. E2 : Ouais et il faut récrire là ?
84. Ppr : Donc le message que vous devez passer c'est quoi ? C'est le résultat ?
85. E2 : Bah, ouais.
86. Ppr : Quel est l'intérêt de passer le résultat ?
87. E2 : Non, il faut qu'on passe toutes les touches pour arriver au résultat.
88. Ppr : Voilà.
89. E2 : Il y en a un paquet ah ?
90. Ppr : Ouais. Donc, il faut vous dépêcher un petit peu.
91. E2 : Il faut retaper toutes les touches qu'on a tapé.
92. E1 : Ouais.
93. E1 : Il faut qu'on remonte au début du calcul. Vas-y.
94. E2 : Vas-y. Je vais marquer tous ce que tu tapes.
95. E1 : Bah, passe moi la feuille.
96. E2 : Voilà et fais vite. Attends, attends. Car s'il faut écrire le 1, le 2 ça ? Il faut écrire chaque fois chaque touche ? (et ils écoutent ce que dit l'enseignant au binôme à côté)
97. Ppr : D'accord ? Donc il faut leur donner la succession des touches.
98. E2 : Il y en a pour 3 ans là si on écrit des chiffres, des points etc.
99. E1 : Je sais pas mais. Monsieur. Quand on a 1,257 il faut écrire la touche 1 puis point puis 2 comme ça ?
100. Ppr : Vous n'allez pas l'encadrer chaque fois. Pour la touche 1, vous écrivez 1, ah ? Car sinon, vous en avez pour des heures.
101. E2 : Mais tape ça aussi 1 virgule 257.
102. E1 : Mais non, il faut écrire chaque fois la touche.
103. Ppr : Non, non, tu tapes 1…C'est 1 point machin et ça correspond à 1a touche 1 et la touche point. D'accord ? Sinon, t'en as pour un petit moment là.
104. Ppu : Allez dans 2 minutes, on échange les feuilles.
105. E2 : Tu tapes le nombre direct. Egale et près tu fais un point…
106. E1 : Point virgule là ?
107. E2 : Je sais pas. Tu mets un espace. Et puis Ans (…) Et après égale.
108. E1 : Egale Ans ?
109. E2 : Non, non. Après on va faire 31 virgule 2. On va faire ça divisé par Ans.
110. E1 : Il faut pas mettre le égale alors.
111. E2 : Non, après tu mets ça divisé. On va faire ça divisé par Ans.
112. E1 : Alors c'est Ans divisé par ça ?
113. E2 : On peut pas car là, il y a une multiplication. Tu peux pas.
114. E1 : Comment tu fais. Il faut noter le nombre.
115. E2 : Mais eux, ils peuvent pas noter le nombre. Ils ont pas de…ils ont pas de stylo.
116. E1 : C'est merveilleux.
117. E2 : C'est chien.
118. E1 : Y a un truc à écrire avec les mémoires mais on sais pas utiliser…Il faut que je marque ce que tu marques ?
119. E2 : On trouve 31 virgule. C'est pas ça. Ils ont pas trouvé ça.
120. E1 : Mais il faut qu'on le note le machin. On peut pas les mettre en mémoire ?
121. E2 : Mais on peut pas. Eux ils peuvent pas écrire.
122. Ppr : Ils peuvent pas écrire ?
123. E1 : Puis que on a trouvé ces nombres. On les a marquer à côté puis on a refait un calcul.
124. Ppr : Mais ça, ça c'est votre brouillon et ensuite vous donnez ce qu'il y a à taper.
125. E1 : Mais s'il y a quelques choses à noter à côté ? Ou à retenir ?
126. Ppr : Vous ne pouvez donner que les touches de la calculatrice.
127. E1 : Bah, vas-y. C'est quoi //retient un nombre ?
128. Ob. (à l'enseignant) : Ils cherchent à utiliser les mémoires depuis le début mais ils ont pas trouvé la solution.
129. E2 : Mais comment on fait ça//
130. E1 : Mais pas encore.
131. Ppr : Vous donnez à votre voisin.
132. Ppu : Allez on fait l'échange même si vous n'avez pas totalement fini.
133. E1 : Mais on pas fini du tout.

134. Ppu : On fait l'échange. Donc, Jonathan, Stéphane avec leur voisine. Nicolas, vous passez à Gladys et Elodie.
135. E2 (au binôme récepteur) : Vous allez beaucoup comprendre ah ?
136. E1 : C'est normal qu'ils vont pas trouver. On a même pas écrit la moitié du début.
137. E2 : Mais tiens.
138. Ppu : Tous les quatre, vous vous échangé entre vous. Voilà. Et puis les deux derniers groupes. Maude et Laura, vous passer à Eve et à Amandine.

Ce que ce binôme a écrit :

1.257 = Ans × Ans × Ans × 4 – 9 × 1.257 + 21 =

Ce que ce binôme a écrit sur leur brouillon :

17,6314837 ; - 6
y = 1,41051891
z = 1,15931225

Ce que exécute le binôme récepteur :

1,257 = ;
Ans×Ans×Ans×4−9×1.257 = ;
L'écran obtenu
- 3,368513628

Le message du binôme récepteur :

① 1,254 = Ans × Ans × Ans × 2 + 15 × 1,254 − 5 =
② 32,1 × 1,254 − 27,7534 =
③ $\frac{17,75387013}{Ans}$ =
④ Ans × Ans − 5 × Ans + 7 = z

139. Ppu : Celui qui, ceux qui reçoivent la feuille marquent bien nom et binôme dans le récepteur. Vous voyez, il y a un trait, il faut marquer votre nom, ah ?
140. E2 (ce binôme commence à exécuter le message) : Egale, Ans, fois Ans fois 2 plus 15 fois 1 point 254 moins 5 égale. Ouais. Ensuite. Mais eux ils demandent de noter ça aussi. On a pas stylo. N'est-ce pas ! On a pas allé loin là. Bon vas-y. Tu mets AC maintenant. (au binôme récepteur) Mais non, le truc qu'on vous a pas donné. Il y pas la fin. Bon mets AC.
141. Ppr : Vous faite ce qu'ils ont tapé. Vous marquez…On vous a donné la fiche navette. Vous avez marqué votre nom dessus ?
142. E2 : Mais ils iront pas loin avec la fiche navette ah. On a près que rien à remarquer, nous.
143. Ppr : Ca fait partie de l'exercice. Ne vous inquiétez pas…
144. Ppu : Voilà, vous avez deux ou trois minutes.
145. E2 : …13 divisé par Ans égale. Ans, Ans moins 5 fois Ans plus 7. (il lit le résultat obtenu) 1 point 91 57 31.
146. E (du binôme à côté) //
147. E2 : Quoi ? (…) Et on marque Ans fois Ans on marque juste le résultat ?
148. E1 : On marque juste le résultat… C'est merveilleux.
149. Ob. : Vous avez vu le problème ? Les mémoires il faut stocker.
150. E2 : Ouais.
151. Ob. : Vous n'avez pas une touche qui est presque stocker ?
152. E2 : Oui. Mais il faut faire comment avec ces touches ?
153. Ob. : Vous essayez Soit vous mettez le nombre et vous mettez Sto A, soit...
154. Ppr : Vous avez fini ?
155. E2 : Sto A. Ouais, on a finit.
156. Ppr : Joana, non Jonathan et Stéphane, vous rendez la feuille…
157. E2 : Sto A ?

Ce que exécute ce binôme suivant le message reçu :

1.254 = ; Ans × Ans × Ans × 2 + 15 × 1.254 − 5 ;
32.1 × 1.254 − 27.7534 = ;
17.75387013 ÷ Ans ;
Ans × Ans − 5 × Ans + 7 = ;

L'écran obtenu :

1.915731337

158. Ppr : Et vous pouvez, vous pouvez maintenant bah, revoir votre message pour essayer d'en écrire un, valable.
159. Ob. : Voilà. Vous allez mettre la valeur dans A. Vous tapez. Non. Vous tapez d'abord la valeur et puis vous mettez Sto A.
160. E2 : Oui. Ah, d'accord.
161. Ob. Essaie avec un petit nombre pour voir si ça marche. Si vous voulez savoir ce qui dans la mémoire…Sto… Vous tapez Rcl. Vous avez jamais utilisé ça dans votre calculatrice ?
162. E2 : Non.
163. Ob. : Ca y est dans votre calculatrice ah ?
164. E2 : Ouais, ouais mais on l'a jamais utilisé.
165. E1 : Et après quand on met A c'est le chiffre qu'on a mis.
166. Ob. : Voilà.
167. E1 : Et fois 2 par exemple.
168. Ob. : Voilà et il contient, il contient le nombre. Bon, voilà, je ne parle plus ah. Débrouillez vous. Parce que, comme vous avez fracassé avec ce problème depuis le début…
169. E2 : Ouis, ça me perturbe. Mais.

Ce qu'ils ont essayé avec l'Ob. :

1,254 Sto A ; A × 2 = ;

170. E2 : Et si on récrivait un message et on mettait en mémoire des trucs ?
171. E1 : Ouais.
172. E2 : Mais c'est super. Vas-y marque c'est bon.
173. E1 : C'est bon ? On refait tout.
174. E2 : Mais donne moi le brouillon et après je recopie. Ca fait…
175. E1 : Tu fais quoi là ? Dis moi avant que tu fasses le stockage ah ?
176. E2 : J'ai fait le stockage.
177. E1 : Mais bah il faut que je marque moi.
178. E2 : Stocker. Tu fais 1 point 257 Sto. A. Et après tu mets ON, AC/ON un truc comme ça et tu fais 4 fois A fois A fois A moins 9 fois A…

179. Ppu : Voilà, vous avez cinq, cinq gros minutes pour réviser votre message, le corriger. Vous ne le redonnez pas au voisin, ah. Allez.
180. E2 : Plus 21 et égale et tu fais Ans Sto B.
181. E1 : Ouais.
182. E2 : Ca c'est le 1er calcul. Tu marques 1 et tu fais 2.
183. E1 : Ouais. 31 virgule 2 Et tu vas mettre en C ça ?
184. E2 : 31 virgule 2.
185. E1 : Et tu vas mettre en C ?
186. E2 (il rit) : 31 point 2 fois A moins 26 point 7184 est égale, est égale. 12 virgule 5. Et après tu mets B.
187. E1 : T'as pas stocké ?
188. E2 : Non, oui j'ai stocké. C'est bon. B divisé par…
189. E1 : Egale 12 virgule 5. Je fais quoi ? Je fais Sto ? 12 virgule 5 ? Egale ?
190. E2 : Oui, ah non, non. Tu fais pas ça. Tu fais rien. Après tu fais 4 et puis B divisé par Ans.
191. E1 : Ouais.
192. E2 : Egale et après on stocke ça. Ca c'est y. Donc. Egale. Ans Sto C.
193. E1 : C. Ans Sto C. Ouais ensuite.
194. E2 : Et après on fait C…
195. E1 : y c'est pas C ?
196. E2 : Si, si. C fois C moins 7 fois C.
197. E1, E2 : Plus 9. Egale
198. E2 : Egale. Ouais c'est ça.
199. E1 : Il faut remarquer ça ?
200. E2 : // C'est le nouveau message que tu fais là ?
201. E1 : Oui.

Ce qu'ils ont tapé jusque là :

1.257 Sto A ;
4 × A × A × A − 9 × A + 21 = Ans Sto B ;
31.2 × A − 26.7184 = ; B ÷ Ans = ; Ans Sto C ;
C × C − 7 × C + 9 = ;

202. E1 : Tu veux que je remarque ?
203. E2 : Egale y, non égale z (…) (au binôme à côté) Il faut d'abord marquer le chiffre et puis Sto.
204. Ppu : Bon, on va arrêter là.
205. E1 : Voilà.

2e message :

1) 1.257 Sto A AC ;
2) 4 × A × A × A − 9 × A + 21 = Ans Sto B ;
3) 31.2 × A − 26.7184 = ; B ÷ Ans = ; Ans Sto C ;
4) C × C − 7 × C + 9 = ;

206. Ppu : Bien, Caroline, tu veux bien passer au, à l'ordinateur. Vous pouvez passer tout les deux si vous voulez, nous montrer ce que vous avez fait. Je crois que j'ai pas passé l'exercice 2.
207. Ppu : Pardon ? Oui, oui, vous prenez votre message. Pour venir le taper. Allez vite par là. Les autres, vous arrêter d'écrire. On va simplement regarder ce que font Caroline et Johanna. Nicolas et Ophélya, vous regardez aussi par là. Stéphane. Alors, déjà j'ai la 1ère remarque c'est quelle utilisent la touche Ans (La classe observe le binôme) (…)

Leur procédé :

1,254 = ;
Ans × Ans × Ans = ; Ans × 2 ; Ans + 5 × 1,254 – 5 = ;
Ans ÷ 32,1 × 1,254 – 27,7534 = ;

208. Ppu : Voilà, vous allez arrêter là. Stop. Qu'est ce que vous en pensez là. Je crois qu'il y a de petits soucis avant, en fait sur l'utilisation de la mémoire mais comme c'est passé trop vite. On ne va pas pouvoir revenir là-dessus. Mais là, sur ce qu'elles ont fait. Vous avez pas ces valeurs là. Vous avez pas forcement exactement ces valeurs là mais sur chacune de vos consigne, la structure est la même et là, je pense qu'il y a quelque chose à dire.
209. E2 : C'est pas bon.
210. Ppu : Pourquoi c'est pas bon ?
211. E2 : Parce que là ils veulent diviser tout après Ans…
212. Ppu : Ils veulent diviser le résultat qu'ils ont trouvé avant…
213. E2 : Par tout calcul, toute la ligne après.
214. Ppu : Par tout ce qui est derrière, normalement.
215. E2 : Et là ça fait 32 virgule 2 divisé par ça.
216. Ppu : Vous vouliez diviser par tout ça, là. Et ça, quand on tape tout ça, on divise uniquement…
217. E2 : 32 virgule 1.
218. Ppu : Par 32 virgule 1. D'accord ? Alors, le problème, il vient d'où. Il vient du fait que sur la calculatrice qu'on vous a donnée, il y a pas de…
219. Es : Parenthèse.
220. Ppu : Y a pas de parenthèses donc qu'il aurait fallu trouvé un moyen. Je crois qu'il y en a qui ont trouvé. Trouver un moyen pour contourner cette absence de parenthèse là. Est-ce qu'il y en qui peuvent donner. Oui, Elodie.
221. E : On a déjà fait le calcul qui est en dessous quoi et après…
222. Ppu : Vous avez fait à l'avance le calcul qui est en dessous.
223. E : Et après on a noté le calcul…
224. Ppu : Donc, vous avez calculé le dessous, vous l'avez le noté.
225. E : Non, le dessous on l'a pas noté. On a déjà calculé le dessus, on a noté le résultat.
226. Ppu : Donc, le dessus, vous avez calculé, vous l'avez noté.
227. E : Le dessous on a calculé et on fait dessus divisé par Ans
228. Ppu : Le dessous, vous l'avez calculé, d'accord ? Vous avez fait égale. Vous avez tapé le dessus qui était noté, divisé par Ans. Vous avez contourné l'absence de parenthèse en utilisant la mémoire Ans. Donc on va voir en classe entière d'une autre façon plus rapide en utilisant les

mémoires pour faire ce calcul sans utiliser du tout, les parenthèses.
229. E : On a fait.
230. Ppu : Pardon. Vous avez fait à la fin ?
231. E : Au début.
232. Ppu : Au début, qu'est-ce que vous avez fait.
233. E : On a noté 1,254 en mémoire A //
234. Ppu : C'est-à-dire vous avez fait comment ?
235. E : 1 virgule deux cent cinquante quatre Sto A.
236. Ppu : Pourquoi vous n'avez pas utilisé plus après, alors ?
237. E : On a pas eu de temps de.
238. Ppu : Donc ça c'était à la fin que vous l'avez fait. Pas au début.
239. E : On a vu sur la fin qu'on peut utiliser les mémoires//
240. Ppu : D'accord. Donc vous voyez, éventuellement, on peut peut-être utilisé les lettres A, B, C, les mémoires A, B, C pour aussi raccourcir le message, le calcul à faire. Bon, on verra ça en classe entière. On essaie de le faire correctement car on garde la calculatrice on peut refaire tranquillement. Alors ce que je vous demande, c'est fermer les calculatrices. Vous fermez juste la calculatrice. Vous fermez que la calculatrice. Le reste, vous laissez sur les tables. Laissez l'ordinateur, sinon. Les feuilles, vous les laissez sur les tables, on va les ramasser. Au revoir.

III. Des copies de binômes

III.1. Copie du binôme la classe 2nd F1 : un rapport « machine arithmétique » à Alpro qui peut se maintenir d'Alpro « machine arithmétique » à Alpro « machine à mémoire Ans »

Exercice 2. Travail en binôme avec la calculatrice Alpro

Feuille navette du binôme : MEURA ; NOVELLO

Phase 1 (10 minutes) : Écriture d'un message

Message pour calculer la valeur numérique de z :

× 1,256 × 1,256 × 1,256 + 15 × 1,256 − 5 ÷ 32,1 × 1,256 − 27,7534 ×
× 1,256 × 1,256 × 1,256 + 15 × 1,256 − 5 ÷ 32,1 × 1,256 − 27,7534 − 5 ×
× 1,256 × 1,256 × 1,256 + 15 × 1,256 − 5 ÷ 32,1 × 1,256 − 27,7534 + 7

Phase 2 (5 minutes) : Exécution d'un message

Noms du binôme récepteur : [illegible]

Écran de la calculatrice : − 14,89406 7

Phase 3 (10 minutes) : Réécriture d'un message

Nouveau message pour calculer la valeur numérique de z :

Aucune modification.

III.2. Copie du binôme de la classe 1er V : d'Alpro « machine arithmétique » à Alpro « machine à mémoire Ans »

Mai Tất Đạt

Phiếu trao đổi

Bài tập 2 (25 phút). Làm việc theo nhóm, dùng máy tính Alpro
Họ tên các học sinh của nhóm :

Pha 1 (10 phút) : Viết thông báo

Thông báo để tính giá trị của z :

1,257 = AC/ON

4×Ans×Ans×Ans − 9×Ans +21 = : 12,5 = AC/ON

Ans×Ans − 7×Ans + 9 =

Pha 2 (5 phút) : Thực hiện thông báo đã viết ở trên
Họ tên các học sinh của nhóm nhận thông báo :

Nguyễn Tuấn Sơn – Bùi Ngọc Tú

Màn hình máy tính Alpro

Ans × Ans − 7 × Ans + 9
1,115 951226

Pha 3 (10 phút) : Viết lại thông báo

Thông báo mới để tính giá trị của z :

III.3. Copie du binôme la classe 2nd F2 : de la machine arithmétique à la machine à mémoire A, B, C et Ans (mémoire variables mathématiques) : une évolution suscitée

Exercice 2. Travail en binôme avec la calculatrice Alpro

Feuille navette du binôme : MONTANA, TABEUTE

Phase 1 ***(10 minutes)*** *: Écriture d'un message*

Message pour calculer la valeur numérique de z : ~~1/.2~~

257 = Ans × Ans × Ans × 4 − 9 × 1.257 + 21 =

Phase 2 ***(5 minutes)*** *: Exécution d'un message*

Noms du binôme récepteur : RAPOSO / ROUDAUD / PORTIER

Écran de la calculatrice

−3,368513628

Phase 3 ***(10 minutes)*** *: Réécriture d'un message*

Nouveau message pour calculer la valeur numérique de z :

1) 1.257 STO A AC

2) 4 × A × A × A − 9 × A + 21 = ANS STO B

3) 31.2 × A − 26.7184 =

4) B ÷ Ans = Ans STO C

5) C × C − 7 × C + 9 = Z

III.4. Copie d'un binôme la classe 1er V : résistance de la mémoire Ans

Ng Thuê Anh

Phiếu trao đổi

Bài tập 2 (25 phút). Làm việc theo nhóm, dùng máy tính Alpro
Họ tên các học sinh của nhóm :

Pha 1 (10 phút) : Viết thông báo

Thông báo để tính giá trị của z :

Bấm 1,257 = Ans . Ấn AC/ON.
Ấn Ans x Ans x Ans x 4 - Ans x 9 + 21 = 17,63148637
Ấn Ans x 31,2 - 26,7184 = 12,5
Ấn AC. Ấn 17,63148637 ấn ÷ ấn 12,5 ấn =
1,410518891.
Ấn = ấn Ans, ấn AC/ón.
Ấn Ans x Ans - Ans x 7 + 9 ấn = 1,115931225.

Pha 2 (5 phút) : Thực hiện thông báo đã viết ở trên
Họ tên các học sinh của nhóm nhận thông báo : Trần Hoàng Mai - Nguyễn Thu Hường

Màn hình máy tính Alpro 1,11593 1225.

Pha 3 (10 phút) : Viết lại thông báo

Thông báo mới để tính giá trị của z :

K° cần bổ sung

III.5. Copie d'un binôme la classe 2nd V : de la mémoire Ans aux mémoires variables

Phiếu trao đổi

Bài tập 2 (25 phút). Làm việc theo nhóm, dùng máy tính Alpro
Họ tên các học sinh của nhóm : Vũ Khánh Vy
Ngô Thị Kim Ngân

Pha 1 (10 phút) : Viết thông báo

Thông báo để tính giá trị của z :

1. Nhập x = 1,257 rồi nhấn phím =: Ans (1).
2. Lấy 4 × Ans(1) × Ans(1) × Ans(1) − 9 × Ans(1) + 21 = . Ans (mới) . (2).
3. Ans (mới)(2) : 31,2 × 1,257 − 26,7184 = . y = Ans (3).
4. Ans(3) × Ans(3) − 7 × Ans(3) + 9 = . z .

Pha 2 (5 phút) : Thực hiện thông báo đã viết ở trên
Họ tên các học sinh của nhóm nhận thông báo : Uyên - Toàn

Màn hình máy tính Alpro

Ans × Ans − 7 × Ans + 9
936,8488861
~~867,475×827~~

Pha 3 (10 phút) : Viết lại thông báo

Thông báo mới để tính giá trị của z :

1. Nhập x = 1,257 rồi nhấn [STO] [A] . [AC]
2. 4 × A × A × A − 9 × A + 21 = [STO] [B] [AC] .
3. 31,2 × A − 26,7184 = . [STO] [C] [AC] .
4. B : C = Ans .
5. Ans × Ans − 7 . Ans + 9 = z .

z = 1,115931226 .

III.6. Copie d'un binôme la classe 2nd F2 : usage des mémoires variables

Exercice 2. Travail en binôme avec la calculatrice Alpro

Feuille navette du binôme : BOISSET et gridel

Phase 1 ***(10 minutes)*** *: Écriture d'un message*

Message pour calculer la valeur numérique de z :

1,254 STO A ON
2 × A × A × A + 15 × A − 5 STO B ON
32,1 × A − 27,7534 STO C ON
B/.C × B/.C − 5 × B/.C + 7 =

Phase 2 ***(5 minutes)*** *: Exécution d'un message*

Noms du binôme récepteur : LEBON et DELPHIN

Écran de la calculatrice

1,915731338

Phase 3 ***(10 minutes)*** *: Réécriture d'un message*

Nouveau message pour calculer la valeur numérique de z :

1,254 STO A ON
2 × A × A × A + 15 × A − 5 STO B ON
32,1 × A − 27,7534 STO C ON
B %. C STO A ON
A × A − 5 × A + 7 =

Annexe des chapitres D2, D3 et D4

I. Des protocoles

I. 1. Binôme de la classe 1er F : des mémoires variables aux mémoires effaçables

Les réponses de deux élèves E1 (avec Alpro) et E2 (calculatrice personnelle) à la situation 1 :
E1 : Réponse 1 : 23.872762 ; Réponse 2 : 296.888424 ; Nombre de fois : 79 ;
Procédure sur Alpro :
1.3141 + 2 × 3.141 × 3.141 Ans Ans
AC/On 3.141 × 3.141 × 2 + 3.141 + 1 =
8 × 3.141759682 × 3.141759682 × 3.141759682 + 6 × 3.141759682 × 3.141759682 – 3 × 3.141759682 – 1 =
E2 : Réponse 1 : 23.872762 ; Réponse 2 : 296.888424 ; Nombre de fois : 79 ;

1. Ppu : Maintenant le processus est le suivant. Vous vous débarrassez de votre calculatrice personnelle et vous travaillez en binôme. Vous ne travaillez que sur Alpro. Vous décidez entre vous qui prenez la commande (il distribue les fiches)…Alors vous voyez, je vous laisse un tout petit peu pour découvrir les consignes et je vous donnerai des explications supplémentaires… Donc, dans un premier temps, vous, vous regardez les, les comment dire, les calculs qu'on vous demandera de réaliser. Et ce qu'on vous demande c'est un message. C'est d'écrire un message. Alors ce message, il est écrit à quelqu'un qui va le lire et ce message ne comprend que des touches de la calculatrice Alpro, que des touches de la calculatrice Alpro. Il est écrit à quelqu'un qui va lire ce message et qui va essayer, grâce à votre message, de comprendre ce que vous lui dites, d'accord ? Et de réaliser le calcul que vous lui avez donné. Bien. Donc vous avez de la place pour marquer le message. Dès que vous écrivez le message, le message partira à un récepteur. D'accord ? Donc je vous laisse du temps pour écrire le message…
2. Ppu : Et vous avez droit à des brouillons. Brouillons, d'accord (il donne une feuille de brouillon à chaque binôme) ?
3. E1 : Ca c'est B, par exemple, B sur C.
4. E2 : Non.
5. E1 : On a trois mémoires A, B et C. Ca c'est déjà A puis B et après c'est C. Et sur…
6. E2 : Et B divisé par C.
7. E1 : On a déjà utilisé A, B et C.
8. E2 : B fois B.
9. E1 : A et B et l'ensemble dans C.
10. E2 : Comment tu dis l'ensemble ?
11. E1 : Tu tapes ça et après tu divises par…
12. E2 : Je te donne le brouillon.
13. E1 (commence à écrire dans le brouillon) : On a déjà utilisé A et B. Et B, C fois C plus…(il montre ce qu'il a écrit à E2). C'est moi qui bosse…Il y a des trucs au-dessus.

Au brouillon :
2 × A × A × A + 15 × A – 5 Sto B
B ÷ 32.1 A – 27.7334 Sto C
C × C – 5C + 7

14. E2 : Ca va ton truc ?
15. E1 : Je sais pas.
16. Ppu : Je vous redis que le message va être lu par quelqu'un qui va être dans le même état que vous, c'est-à-dire qui ne dispose que d'un stylo et de la calculatrice Alpro, pardon et qu'il ne disposera que de la calculatrice Alpro. Je vous répète, vous, vous avez un stylo mais lui il ne va pas avoir le stylo, il va avoir que la calculatrice Alpro…
17. E1 : Ca a l'air marché. Je sais pas si tu remplaces, tu mais…
18. E2 (il essaie le message sur sa calculatrice).

Leur message :
Tappe : 1.254 Sto A
2 × A × A × A + 15 × A – 5 Sto B
B ÷ 32.1 – 27.7534 Sto C
C × C – 5 × C + 7 =

19. Ppu : Pardon ? Oui, je crois qu'on va attaquer à la 2e phase. C'est-à-dire que vous avez écrit votre message. Ca y est ? Tout le monde a écrit son message ? Donc il suffit maintenant de les renvoyer. Je vous propose la procédure suivante. Vous donnez, échangez avec votre voisin à droite. Non vous me regardez. Voyons, les deux binômes qui sont là, vous échangez vos messages. D'accord ? Les deux binômes qui sont ici, ici là, vous échangez vos messages. Et les deux binômes qui sont au fond, vous échangez vos messages. Et vous échangez comme ça. Alors, celui qui reçoit maintenant. Va exécuter. Oui ? Bon, il doit constater que le message ne comporte que les touches d'Alpro, il utilise la calculatrice Alpro et il obtient quelques choses à l'écran et donc il se contente de lire le message, de l'exécuter sur Alpro et de marquer le résultat. Alors éviter quand même les commentaires entre vous là, entre l'émetteur et le récepteur, parce que vous savez que votre récepteur peuvent //. Alors il faut faire vite là.

Le message reçu :

1)* 1.257 → Sto → A
2) 31.2 × A – 26.7184 → Sto B
$\underbrace{4 \times A \times A \times A - 9 \times A + 21 =}_{Ans} \rightarrow \underbrace{Ans / B}_{StoC}$
4) * B = C × C – 7 × C + 9

20. Ppu : Vous marquerez ce qui apparaît à l'écran même ce qu'il apparaît à l'écran est en rouge. Chut…L'émetteur est indépendant du récepteur…
Ce que ce binôme a tapé suivant le message du binôme récepteur :
1,257 Sto A
31.2 A – 26.7184 Sto B
AC/ON
1.257 Sto A
31.2 A – 26.7184 Sto B
1.257 Sto A
31.2 × A – 26.7184 Sto B
4 × A × A × A – 9 × A + 21 =
Ans ÷ B Sto C
C × C – 7 × C + 9 =
Le résultat obtenu :
1.115931226

21. Ppu : C'est bon, alors, tout de suite, retour à l'envoyeur ah ? Ce qui est bien c'est que le récepteur marque son nom. Et le récepteur va envoyer le message à son émetteur. Ca y est ? Vous renvoyez à l'émetteur ? Et puis il y aura des surprises, l'émetteur se dit « Oh, là la, le récepteur m'a mal lu, n' a pas compris. Oui. D'accord ? Et quand vous dites ça, vous vous mettez dans la peau de récepteur. Mais ce qui est intéressant est que vous vous mettiez sur vous et vous vous dites « qu'est ce que je dois changer dans le message pour que le récepteur ne se trempe pas mon intention. Donc vous êtes invité à rectifier si vous pensez que c'est nécessaire…sinon vous faites rien.
22. E1 (ils sont entrain de taper le message de l'autre binôme) : Tout à l'heure j'ai essayé mais ça marche pas.
Ce que le binôme récepteur a tapé suivant leur message :
1.254 Sto A
2 × A × A × A + 15 × A – 5 Sto B
B ÷ 32.1 – 27.7534 Sto C
C × C – 5 × C + 7 =

Le résultat de leur message exécuté par l'autre binôme : 874.5339988

23. E1 : Tu mets égale même qu'ils n'ont pas mis égale.
24. E2 (au binôme à côté) : Il faut que tu mettes égale.
25. E (du binôme à côté) : Mais ce n'est pas la peine.
26. E1 : Tant pis pour eux. Tu fais une expérience et ils en seront d'accord.
27. E2 : Et voilà, tu vois.
28. E : Oui j'ai mis égale alors. (Et il ajoute le signe égale dans leur message).
29. Ppu : Ca y est ? C'est bon ? Je peux ramasser ? Pensez à y mettre votre nom. Je ramasse.
30. Ppr (au binôme récepteur) : Il y a une erreur ? Mais qui a écrit ce message ?
31. E (du binôme récepteur) : C'est à côté.
32. Ppr : Ah, c'est le leur ?Non, non par ce que vous avez pas remarqué ce que vous avez obtenu.
33. Ppu : Quoi que, vous n'avez pas souvent fait des messages mais il est bon ça…C'est bon ? Je peux ramasser ? Je peux ? Et vous voyez que l'enjeu ici c'était de faire le message et qui, qui utilise les mémoires Ans et A. Je regarde les messages et je vois toute sorte de choses…(la sonnerie sonne) Alors on va simplement, on va prendre le temps pour regarder un message…(il fouille les messages ramassés). Voilà, je vais demander à C. Ah, c'est encore vous ? Donc je vais peut-être changer. Bon, je vais demander à Long. Voilà, de venir ici nous exposer le message. (le binôme vient au tableau) Allons-y. Je peux récupérer toutes les feuilles ? (à ces deux groupes qui discutent sur la présence ou non de signe « égale ») vous ne battez pas pour les maths ah ? Vous voyez qu'il a stocké 1.254 dans la mémoire A. Alors, ça mérite de commenter. Qu'est ce qu'il est entrain de faire ?
34. E (du binôme au tableau à celui) : Non, c'est pas ça.
35. Ppu : Ah, si en plus s'il fait pas ce qu'il a dit.
36. E : Non, c'est 7534 mais pas 6534.
37. Ppu : Alors. D'accord. Qu'est ce que vous avez fait ? Qu'est ce qui s'est passé ? Qu'est ce que vous pouvez faire ? Mais comme ça, vous allez effacer tout ce que vous avez fait. Ah, on recommence. Peut-être que non. Ecoutez moi bien. Ce qui est dans la mémoire A est toujours dans la mémoire A.
38. E (du binôme mais qui tape sur l'ordinateur) On recommence avec la mémoire B simplement.
39. Ppu : Très bien, très bien. La mémoire A n'a pas bougé. Il veut simplement changer le contenu de la mémoire A.
40. Ppu (au binôme) : Qu'est ce qui s'est passé ? Qu'est-ce que vous avez fait ?
41. E (du binôme au tableau) : Par ce qu'ici il y a pas de parenthèse pour utiliser donc on…
42. Ppu : Donc vous avez décidé de mettre le dénominateur de la fraction dans la mémoire B. D'accord...Donc là. Vous pouvez commenter ?
43. E (du binôme au tableau) : Donc là, c'est, c'est…on a calculé y.
44. Ppu : Vous calculez y. Vous voulez plutôt calculer le numérateur et vous avez divisé par le numérateur pour calculer y. D'accord ? Et vous avez utilisé la mémoire Ans pour le dénominateur.
45. E (du binôme au tableau qui tape) : C'est Ans Sto C.
46. Ppu : Mais est ce qu'il y a une erreur là. A cause de Ans ? Parce que le Ans a été changé.

47. E : Ah, oui.
48. Ppu : Il faut reprendre. Ah oui, il faut rester très vigilent sur le statut des mémoires Ans, A, B, C.
49. E (qui tape) : A fois…
50. Ppu : Ca c'est le dénominateur, d'accord ? Egale d'accord ?
51. E (le 2e du binôme au tableau) Divisé par…
52. Ppu : Divisé par B…Egale. Et Ans Sto C. Donc le C c'est quoi alors là? Le C c'est y. D'accord ?
53. Es : 5 fois C plus 7.
54. Ppu : Plus 7. Pour ceux qui calculent avec x = 1.257 c'est la valeur de z, ils devraient connaître le résultat.

Ce que tape ce binôme :

1.254 Sto A
32.1 × A – 27.7354 Sto B
AC/ON
32.1 × A – 27.7534 Sto B
2 × A × A × A + 15 × A – 5 =
Ans ÷ B =
Ans Sto C
AC/ON
2 × A × A × A + 15 × A – 5 =
Ans ÷ B =
Ans Sto C
C × C – 5 × C + 7 =

55. Ppu : Voilà, on va faire une toute petite pause s'il vous plaît. Je ne sais pas quelles sont vos habitues mais une toute petite pause, s'il vous plaît, mais ça sera bien que vous sortiez pas mais vous détendiez un tout petit peu. Moi aussi, d'ailleurs. Il me semble qu'il y a des choses sur l'ordinateur que vous pouvez regarder découvrire…

Le binôme n'a pas proposé un autre message.

56. Ppu (il distribue les fiches de la situation (Alpro, Calculator 1) : N'oubliez pas Alpro. La calculatrice Alpro est en mesure de faire tous les calculs que vous faites. Il y en a qui l'aiment pas beaucoup, pour l'instant (…) (le temps que les élèves répondent aux questions de l'exercice).
57. Ppu : Je peux ramasser ? Je vois qu'il y a des feuilles qui ne sont pas encore remplies. Vous avez marqué votre nom et prénom. Je peux prendre ? Merci. Je ramasse tout maintenant. Je vais corriger(…) Venez par là (il dit aux élèves qui ne peuvent pas voir le tableau). Je peux pas vous laisser comme ça, sans que vous voient pas. Donc nous sommes tout d'accord pour dire que l'écart est 0.2. Et après sur la 2e question, il y a des choses différentes mais je vais prendre par exemple cette réponse-là. (Il commence à écrire au tableau en disant). Que je vois souvent, f de – 2 égale à 5, f de – 1 est égale à 2 et f de 0.6 est égale à 1.36. Ce sont des écritures que je vois assez souvent. Je pense qu'ici on lit la 6e valeur de x est moins 2. Il faut bien sûr se met d'accord que la 1ère valeur de x est – 3, la 2e valeur de x est – 2.98 etc et donc la 6e valeur de x égale - 2. Et que – 2 au carré plus 1 égale 5. La 11e, c'était la 6e, la 11e valeur de x est – 1. Son image est 2. La 13e valeur de x est – 0.6 son image et 1.36. Alors il y en a qui compte la 0 à la place de 1. C'est-à-dire qu'ils ont pas compté la 1ère valeur de x qui est – 3, la première valeur de x. Ils ont donc tout décalé. Voilà, voici les résultats. Je ferai pas plus de commentaires. A mon avis, sauf une petite remarque, sur votre calculatrice vous avez souvent cette touche « ± » là qui signifie que vous faites précéder d'un nombre pour signe moins. Pour les nombres négatifs ce n'est pas nécessaire. Sur la calculatrice Alpro c'est la touche « - » je crois.

Au tableau :

f(- 2) = 5
f(- 1) = 1
f(- 0.6) = 1.36

58. Ppu : Donc maintenant on va passer au 2e exercice. Nous sommes entrés dans le problème, enfin un p'tit problème. Vous savez qu'on a besoin de ce genre lorsque vous faites la représentation graphique d'une fonction. Mais lorsqu'on fait la représentation graphique d'une fonction on a besoin d'un pas nettement beaucoup plus faible que ça. L'idée qui est la suivante : qu'on va diminuer le pas. Mais cette fois-ci, vous allez rentrer votre calculatrice personnelle. Tout à l'heure on vous laissait la calculatrice mais cette fois-ci vous n'avez que la calculatrice Alpro. Donc nous allons vous proposer de travailler avec un pas mais nettement plus faible… (il distribue les feuilles) Je vous laisse d'abord découvrire l'exercice, enfin les différents commentaires…Bien, vous découvrez l'existence d'un robot. Ce robot-là, il peut faire des choses là : il sait appuyer sur des touches de la calculatrice Alpro comme vous savez faire, lui il saura faire. Mais à condition à lui dire de touches. Afin de lui dire, non de lui écrire des touches donc sur le message. Et il sait faire la 2e chose, il sait imprimer automatiquement ce qui est affiché comme résultat sur la calculatrice Alpro après l'exécution du signe « = ». Vous auriez du remarquer que après la touche « = » les résultats s'apparaissent en bleu sur la 3e ligne à droite de la calculatrice Alpro. Vous savez que la calculatrice Alpro comporte 3 lignes à l'écran. La première ligne c'est la ligne d'état, on peut l'appeler comme ça où vous voyez certaines touches, la 2e ligne où vous entrez les données, vos calculs et la 3e ligne les résultats après que vous appuyiez sur les touches égale ou Sto par exemple ou donner des messages. Et donc le Calculator sait imprimer automatiquement ces résultats ce qui est en bas à droite. Donc on décide d'écrire un message à Calculator pour lui faire faire des messages que vous trouverez long. D'accord ? Parce que quand vous diminuez le pas vous êtes amené à augmenter le nombre de calcul. Alors ce message doit imprimer toutes les images de la fonction f. C'est-à-dire, par exemple ici, il doit imprimer 5, 2 ou 1.36, d'accord ?

Si on lui avait donné un pas égale 0.2. Il doit imprimer toutes les images et seulement, voilà toutes les images qui appartiennent à l'intervalle comme tout à l'heure, c'est-à-dire de – 3 à 2 avec la même fonction. Voilà, vous êtes invité à écrire le message à Calculator qui va recevoir le message (…)(le temps que les élèves répondent aux questions).

59. E1 : Qu'est-ce qui s'est passé ? C'est un peu ça. A ça, à ça…On est obligé de faire ça chaque fois.
60. E2 : Comment tu retrouves ça.
61. E1 : Je vais passer à…Je sais pas.

Leur premier message écrit au brouillon :

3 = Ans
Ans × Ans – 1
3.003 = Ans
Ans × Ans – 1

62. E2 : Il y a que du Ans ? Tu fais ça ? (Et il écrit le 2e message que E1 lui dicte).
63. E1 : Je sais pas. x fois x plus 1. Et x plus A fois x plus A.
64. E2 : C'est quoi A ?
65. E1 : C'est l'écart. Et Ans fois Ans.
66. E2 : Et Ans car Ans c'est ça.
67. Ppu : Bon Calculator attend votre message.

Leur 2e message écrit au brouillon :

x × x + 1
(x + A) × (x + A) + 1 = ~~Ans~~
~~(Ans) × (Ans~~
x +

68. E1 : Et Stocké, stocké.
69. E2 : Bah, non car…
70. E1 : En fait, t'as raison. On fait – 3 égale.
71. E2 : Non, mais A c'est 0.03.
72. E1 : Non, attends, A plus 0.03 est mis dans B. Et B fois B moins 1.
73. E2 : Mais pourquoi moins 1 ? C'est plus 1. Tu vois la fonction.
74. E1 : Et après comme on a déjà B, et après ça fait, c'est B…T
75. E2 : Mais tu vois moins 3 c'est x. C'est fou ça. A moins 3.
76. E1 : Et tu fais le tour comme ça. T'as A, on fait le calcul avec B et on augmente B tout ça et on retourne comme ça. Ca fait le tout et on est déjà intelligent.
77. E2 (il n'a pas l'air d'accord) : Le résultat est comme ça.
78. E1 : C'est bon comme ça. T'as pas besoin de recommencer chaque fois.
79. Ppu : Il vous reste une minute pour que vous écriviez le message. Même si vous commencez votre message. Mettez votre nom s'il vous plaît…Et vous écrivez.
80. E2 : Donc c'est x égale A ?

Leur 3e message écrit au brouillon :

- 3 = A
A + 0.03 Sto B
3 B × B + 1 =
B + A 0.03 Sto A
~~A × A + 1 =~~ …

81. Ppu : C'est bon ? Je peux les ramasser, s'il vous plaît ? C'est bon là. Je ramasse quand même.
82. E1 : Si on fait comme ça, il faut pas faire le tour comme ça en passant par - 3…
83. E2 : Mais il faut pas passer. Tu mets juste A égale x.
84. E1 : OK.

Leur message écrit dans la fiche :

x ~~3~~ = A - 3 Sto A
A + 0.03 Sto B
B × B + 1 =
B + 0.03 Sto A

85. Ppu : Je vous propose que c'est moi qui joue le Calculator et vous viendrez ici pour me donner des messages. Voilà je leur redonne le message et mois je suis Calculator. Je sais parfaitement ce que je sais faire, ah ? Donc tout le monde peut voir ? Je vous écoute et je suis Calculator ah ? J'attends les touches.
86. E (du binôme qui est demandé d'aller au tableau) : – 3 Sto A.
87. Ppu : Oui là, je sais faire. Je pense que c'est le moins là. – 3 Sto A AC/ON
88. Ppu : Pour l'instant je sais faire car les touches je sais faire et je comprends. AC/ON.
89. E : Ensuite 0.03 Sto B et AC/ON.
90. Ppu : AC/ON pour l'instant ce sont des touches et moi, je comprends.
91. E : Ensuite A plus la mémoire B et on fait égale.
92. Ppu : Egale. Oui.
93. E : Et sur le résultat on fait AC/ON et on fait Ans fois Ans plus 1.
94. Ppu : OK. Vous avez suivi un peu les opérations là ? Ans fois Ans et plus 1. Je fais quoi là ? Je fais simplement. Ah, voilà, Ans fois Ans plus 1. Egale et puis ?
95. E : Et après je recommence ça on refais et puis…
96. Ppu : Attends, attendez. Moi, je suis Calculator
97. Es (rient)
98. Ppu : Et je ne comprends que…
99. E : Bah, on fait égale et Stocké.
100. Ppu : Je fais égale là ? Ouis ? Et puis, qu'est-ce que je fais ?
101. E : Non, AC/ON. Et après Calculator il lit le résultat.
102. Ppu : Non, non, je suis Calculator et je sais faire des choses et je l'ai dit.
103. E : Et il a mémorisé le résultat c'est ça ?
104. Ppu : Oui, ça y est. Le résultat est mémorisé dans Alpro.
105. E : Oui, oui. Donc on refait A plus B fois 2.
106. Ppu : Alors A plus B fois 2.
107. E : Et on fait égale. AC/ON. Et donc on refait Ans fois Ans plus 1.
108. Ppu : Oui, on fait égale.

109. E : Et on a le résultat. Et on refais ça, on multiplie par 3. En fait, à chaque fois on multiplie par 3, par 4…
110. Ppu : D'accord, est-ce que vous êtes d'accord que le message est correct ou non ? Et que ça imprime tout ce qui est demandé ? A chaque fois que j'appuie sur la touche égale, j'imprime ça car je suis Calculator, est ce que vous pensez que j'imprime toutes les images et seulement les images ? Oui, ça va ? Le message est correct.
111. Es : Oui.
112. Ppu : Ets ce que le message est un peu long ?
113. Es : Si, si.
114. Ppu : Si, par ce que vous n'avez pas encore écrit totalement tout le message ?
115. E (du binôme) Si, si.
116. Ppu : Ah, non, non. Parce que ce que je veux dire, pour l'instant je vois (il point le doigt dans leur production) un deux trois…une trentaine là. Il n'est pas encore total.
117. E (du binôme) : Parce que je fais n.
118. Ppu : Ah, il n'est pas encore fini le message ? (il continue de taper) Mais n, n c'est quoi ? Mais le Calculator, n qu'est ce que c'est ?
119. E (du binôme) : C'est un nom.
120. Ppu : Non, non, là Calculator il sait appuyer sur les touches.
121. E : Il appuie fois 1, 2, 3, 4, 5, 6, 7, 8, 9 jusqu'à l'infini.
122. Ppu : Jusqu'à l'infini ?
123. Es : Non, non.
124. E (du binôme) : Non jusqu'à la fin de l'intervalle.

Ce que l'enseignant a tapé suivant les instructions du binôme :

3 Sto A
AC/ON
0.03 Sto B
AC/ON
A + B =
AC/ON
Ans × Ans + 1 =
Ans × Ans + 1 =
A + B × 2 =
AC/ON
Ans × Ans + 1 =

125. Ppu : On a quel problème là ?
126. E : La répétition.
127. Ppu : Oui, la répétition et quoi ? Car vous dites il faut aller jusqu'à l'infini et d'autres disent non. Alors, c'est…
128. E : C'est l'intervalle.
129. Ppu : Mais du point de vue du message c'est…
130. E (un autre de la classe) : C'est le problème de finition.
131. Ppu : Pardon c'est…
132. E : Finition.
133. Ppu : Oui, finition ce n'est peut – être pas le bon mot mais c'est le problème, le problème de…le point d'arrêt, non ? Il faut qu'on s'arrête, non ? Bon, on va vous amener à un nouveau problème. Parce que finalement on s'aperçoit d'une chose est que toutes les actions à Calculator sont écrites sous formes de touches. Donc ce qu'on aimerait savoir c'est qu'il y a combien de touches ?
134. Es : C'est long.
135. Ppu : C'est long, ah oui d'accord c'est long mais c'est combien, long ? (il donne la nouvelle fiche à chaque binôme) (…)
136. E1 : Oublie cette partie-là. Mais on commence en fait si A c'est 0 normalement, par exemple.
137. E2 : Non, on commence par moins 3. L'intervalle c'est combien ?
138. Ob. C'est de – 3 à 2.
139. E1 : Ca fait 5 donc c'est comme si tu partais de 0 à 5.
140. E2 : Il y 5 l'intervalle ? Et le pas c'est 0 virgule 0 3.
141. E1 : Mais il y a 5. C'est comme si tu partais de 0 à 5 et donc…avec ce qu'on a déjà fait on a juste à faire…A part ça, à part de 0 et après c'est 0 virgule 0 3 et on compte.
142. E2 : Tu comptes ça le nombre de fois.
143. E1 : Tu fais 5 divisé par 0.03.
144. E2 : Mais on a pas un nombre entier
145. E1 : Ah, ouis ? Tu arrondis, tu arrondis. Au plus petit. Ca fait, égale à 166. Tu comptes le nombre truc tapé et tu fais fois 166 puis…(sur leur bouillon ils ont écrit « $\frac{5}{0.03} \approx 166$ »
146. E2 : Voilà.1, 2…Vas-y, marque dans la fiche.
147. E1 : 1, 2…14, 15. (ils comptent sur le 3e message écrit au brouillon)
148. E2 : Mais non, ça c'est la même chose que là. Ca c'est la redite de ça. (il pointe sur la ligne barrée « A × A + 1 »).
149. E1 : Mais c'est important, c'est à partie de ça qu'on recommence le calcul. Car ici il faut remettre, enfin le x déjà modifié.
150. E2 : Ouais. Ca fait 21. Multiplié par 166.
151. E1 : T'est sûr que c'est 21 (il recompte). 1, 2, 3…21, 22.
152. E2 : Non, ah, bon (il fait le calcul 22 × 166 sur sa calculatrice personnelle), ca fait 3652. Vas-y, tu marques ça.
153. E (du binôme à côté) : Vous avez trouvé combien ?
154. E2 : 3652.
155. E : C'est différent du notre.
156. E1 : Mais comme vous faites la même chose que nous donc c'est probablement différent.
157. Ppu : Voilà je pense que ce n'est pas…La question est que combien de fois Calculator appuie sur les touches c'est-à-dire la longueur du message. Caculator lui il sait appuyer. Il faut bien imaginer ce que c'est le Calculator ah. Vous lui donnez un message qui comporte les touches et lui il appuie sur les touches.
158. E2 (qui écrit la réponse) : 3652 voilà.

Leur explication sur le nombre de fois écrite sur la fiche :

$$\left.\begin{array}{l} A + 0.03 \; Sto\; B \\ B \times B + 1 = \\ B + 0.03 \; Sto\; A \end{array}\right\} 22 \; touches$$

$$\frac{5}{0.03} = 166 \rightarrow 22 \times 166 = [3652]$$

159. E2 : Attends je vérifie.
160. Ppu : Bon je vais ramasser. Encore une minute. Pas plus. Vous voyez (Ca sonne). Attendez, attendez. Vous avez encore 15 secondes. Je vous le dis pas car de toutes façon c'est grand. Vous voyez que vous avez trouvé tous beaucoup. Donc vous vous dites que Calculator n'est peut-être pas encore une machine perfectionnée. Donc dans cette séance on vous demande votre avis. On vous demande votre proposition sur les capacités de Calculator pour que le message soit plus court mais pour que, pourtant qui fasse la même chose mais avec un message plus court. Je répète qu'il fasse la même chose mais en plus court. (Il distribue la nouvelle fiche).
161. E1 : Je vois pas un message plus court que ça.
162. E2 : Mais si, déjà //.
163. E1 : C'est l'intervalle…
164. E2 : Par exemple, on ne passe qu'une fois, au début, tu vois (il pointe la 1ère phrase dans leur message « - 3 = A »)
165. E1 :Non, non ce n'est pas important car ce truc c'est seulement qu'une fois. Tu mets, au début c'est – 3 mais après c'est plus – 3.
166. E2 : Donc c'est plus 4 alors ?
167. E1 : Pour quoi 4 ? Mais ce n'est pas important car c'est le nombre de fois dont on parle. Ce qui est important c'est qu'on parte de 0 à 5. (il explique en utilisant le 3e message écrit au tableau dont les instructions sont numérotées) Tu retombes pas à ① mais à 2 tu vois.
168. E2 : Mais on est déjà à 2.
169. E1 : Normalement suivant la logique, tu fais ça et tu retombe à ① n'est ce pas ? Mais ce qui ne faut pas faire.
170. E2 : Une fois qu'on a fait entrer, fait B plus 0.03 S T O A on fait plus A plus 0. 03 S T O A ?
171. E1 : C'est juste ce que je veux dire, tu comprends. Une passe par là (l'instruction « - 3 = A ») juste une fois au début.
172. E2 : Donc on a passé une fois au début et après non.
173. E1 : Ouais.
174. E2 : Donc on a pas besoin de trouver un truc pour //
175. E1 : Mais Calculator il sait pas faire ça. C'est nous qui doit faire ça. C'est notre problème.
176. E2 : Donc tu vois on passe pas par là (il point la 1ère instruction dans leur message).
177. E1 : Ouais mais il faut pourvoir dire ça à Calculator. Sinon il refait la ligne 1 et on refait toujours la même chose. Qu'est ce que t'as mis dans la fiche ? Tu mets «Pierre c'est toi qui est intelligent c'est toi qui doit …» (il rit)
178. Ce qu'ils ont marqué comme proposition à Calculator : « x2 »
179. Ppu : Nous voulons que le Calculator fasse la même chose mais avec un message nettement plus court. Je parle bien que c'est Calculator. Oui, voilà, pour faire le problème là. Qu'est ce que vous verrez comme capacités supplémentaires à attribuer au Calculator. (Il ramasse les feuilles) Merci.

Les deux premières séances ont terminé

180. Ppu : Installez vous au même endroit. Aux mêmes endroits. Il y en a qui ne marchent pas ? Et qui s'en occupe ? De toute façon, vous rouvrez de nouveau votre calculatrice. Ah, ça a l'aire de marcher. Non ? Mois je fait (il ouvre Alpro sur son ordinateur). Voilà, vous entrez le nouveau exercice, numéro 1, et la nouvelle question. Et la calculatrice est à votre disposition. Pardon ? Il y a un problème ? Quelqu'un est allé chercher l'administrateur ? Oui ? Donc je vous rappelle un petit peu le contexte dans lequel nous sommes. Nous nous sommes intéressés à une fonction et à ses images des variables qu'on s'attribue selon un certain pas dans l'intervalle de – 3 à 2 et avec le pas est de 0.03. Ca y est, tout le monde rappelle le problème mathématique enjeu ? Je vous rappelle aussi que dans la dernière séance nous avons fait un certain nombre de travail. On vous a demandé des propositions pour améliorer le Calculator et j'avais insisté sur le fait que ce robot Calculator n'est pas Alpro. Le robot se sert d'Alpro. Il appuait sur les touches d'Alpro. Il est donc différent d'Alpro. Il a la capacité d'exécuter les touches d'Alpro quand on lui demande et d'imprimer automatiquement. Je vous ai demandé des propositions pour améliorer le robot Calculator en tenant comte du fait que nous nous sommes aperçus que le message adressé à Calculator comportant une longueur qui est autant de nombres de touches que Calculator doit appuyer. Et nous l'avons calculé. Je pense que certains, il y a certains qui ont déjà calculé ce nombre. Deux milles et voir quatre milles touches. C'est énorme. Donc un message qui comporte 4 milles touches. On se rend qu'il y a quelque chose à faire. Alors vos propositions, et j'ai lu votre message et je m'aperçois que…D'ailleurs certains n'ont pas marqué dans la précipitation leur nom. Par exemple celui là. (Il lit le message) « que ça décline toutes les images lorsqu'on appuie sur la touche « = » successivement » Qu'est ce qui se rappelle ? Oui, marquez votre nom simplement. Pour qu'il y ait pas des commentaires à faire. Et ceci « savoir des intervalles de fonctions » ? C'est marqué au crayon avec une jolie écriture là. Je sais pas si c'est à une mademoiselle ? « Savoir des intervalles de fonctions » C'est vous ? Que vous remarquiez que les jeunes filles oublient souvent d'écrire leur nom là.

181. Es : (rient)
182. Ppu : Ce que je voulais dire, en regardant les messages je me suis aperçu que beaucoup d'entre vous, les propositions portaient sur Alpro. Or notre demande c'était d'améliorer Calculator. J'ai vu quand même qui ceux qui parlaient de Calculator. Je vois celle là (il lit le message) « Etant donnée que A = 0.03 est toujours x 2 pour chaque image, Calculator devrait s'habituer au bout de 2 calculs ». S'habituer au bout de 2 calculs ? Je me suis dit quand même que ce message s'adressait à Calculator. C'est une capacité à Calculator. Est-ce qu'on peut formuler un peu plus précisément ? Ca veut dire quoi « s'habituer » ?
183. E (de la classe) : C'est savoir répéter les mêmes…
184. Ppu : Les mêmes ?
185. E : Les mêmes, les procédures.
186. Ppu : Les mêmes procédures mais…d'accord ?
187. Es : (silence)
188. Ppu : Est-ce qu'on est d'accord là-dessus ? Ou pas ? (en s'adressant à l'élève qui vient de prendre la parole) C'est vous qui avez fait ça ? Non. Je sais pas mais est-ce que celui qui écrit ça peut nous dire…Est-ce que c'est une bonne traduction ou pas ?
189. E (qui a écrit ça, d'une manière timide) : Qui refait les modifications qui …
190. Ppu (il répète) : Ca va refaire les modifications qui
191. E : ont été faites.
192. Ppu : Qui ont été faites. On sent quand même qu'il y a l'idée de reprendre les procédures, je sais pas quoi, mais de les modifier de telle manière d'assurer la répétition, oui ou pas ? Ou c'est pas ça son idée ? Il faut qu'on en discute un petit peu si…Ouis, assurer la répétition c'est quand même la question de base que vous vous faites à la main. En fait, vous faites bien les répétitions et vous modifiez des choses pour faire la répétition. J'aimerais qu'on penche à cette idée et qu'on sache que s'il y a des choses à répéter, qu'est ce qu'il y a à répéter. Qu'est ce qu'on va demander à Calculator de répéter en n'oubliant pas que l'on va lui demander. D'abord, on aimerait savoir qu'il y aura que Calculator doit répéter. C'est la question à laquelle j'aimerais que vous répondiez. Je vous distribue les petites fiches. Vous disposez toujours et seulement de la calculatrice Alpro. (au binôme dont le poste ne marche pas) Toujours pas avec vous, ah. J'espère que ça sera résolu bientôt. Ah, pas avant 14h ? Alors il faut que vous disposiez d'ailleurs. Vous choisissez votre groupe. Voilà je commente ce qui est marqué. Voilà, on veut assurer la répétition et on sait que cette répétition devrait permettre de diminuer le nombre de touches à demander à Calculator d'appuyer. Car on a déjà vu et on sait que c'est une grand nombre. On décide donc d'améliorer…On vous propose de prévoir que Calculator puisse répéter l'exécution d'un groupe de touches d'Alpro. Oui je redis problème depuis tout à l'heure. C'est…Notre objectif est de mettre en place un groupe de touches d'Alpro que si Calculator les répétait ça permettrait d'obtenir ce qu'on veut obtenir. Je vous rappelle ce qu'on veut obtenir. On veut obtenir toutes les images de la fonction f de x est x au carré plus 1. On vous redonne la fonction. (il écrit au tableau). Je vous redonne la fonction. La fonction est f de x est x2 + 1. On prend x dans l'intervalle, – 3 à 2. Comme première valeur de x est – 3 et on a un pas est égale à 0.03. D'accord ? (en pointant ce qu'il a écrit au tableau) et on veut obtenir toutes les images de, par f, pour les valeurs de x en allant de 0.03 à 0.03 à partir de – 3 dans l'intervalle – 3 à 2 et on se dit qu'il y a des répétitions et on aimerait connaître les touches que Calculator serait amené à répéter.
Au tableau :

$f(x) = x^2 + 1$
x dans [- 3 ; 2]
première valeur de x = - 3
pas = 0.03

193. Ppu : C'est les touches d'Alpros, ah ? Je vais vous distribuer les brouillons, les feuilles de brouillons.
194. E2 : On a A égale x, A plus 0 virgule 0 3 S T O au B, B fois B plus 1.
195. E1 : Non, c'est A égale à moins 3, non ? Et après on fait…
196. E2 : Mais c'est x moins – 3.
197. E1 : - 3 S T O au A, voilà, un truc comme ça.
198. E2 : Mais en tout cas, on a pas besoin de répéter là. Ca c'est au début. Au début on a – 3 S T O au A, d'accord ? (Il commence à écrire dans leur brouillon) Au début on a ça, - 3 S T O au A. D'accord
199. E1 : Mais je suis pas d'accord. Laisse ça pour le moment. Maintenant on a A, c'était au A.
200. E2 : A plus 0 virgule 0, 3.
201. E1 : Et ça S T O au B.
202. E2 : Ouis, c'était au B. On avait fait B fois B plus 1, égale et c'était le résultat. Ensuite on avait fait B plus 0 virgule 0 3 S T O au A.
203. E1 : D'accord.
204. E2 : Et ensuite on revenait, ça fait A fois A plus 1. Encore le résultat et on faisait A plus 0 virgule 0 3 S T O au B.
Leur 1er groupe de touches écrit au brouillon :

A – 3 Sto A A = - 3.003
A + 0.03 → StoB
B × B + 1 = …
B + 0.03 → StoA
A × A + 1 = …
A + 0.03 → StoB

205. E1 : Non, non, ça on répète pas la répétition car il suffit de changer au fait, à chaque fois. Par exemple donc S T O au B.
206. E2 : Il faut savoir répéter ça.
207. E1 : Mais après , regarde tu vois ici on répète plus parce que ici c'est A, ici c'est B. Donc en fait on barre ça et puis on repend B ça, B plus machin S T O au A ici car B ça va être ça au fait.
208. E2 : Mais B ça va être ça. A plus 0.03.

209. E1 : Non, B ça va être ça et après on met ce chiffre-là dans B et on reprend en fait ce chiffre-là dans B au fait et comme ça devient A donc lorsqu'on recommence ici A devient A plus 0. 03, plus encore un 0.03 et comme ça, ça fait une répétition. En fait tu vois, on barre ça et d'ici tout de suite on peut retourner et ça marche…Il faut juste.
Leur nouveau groupe de touches après les modifications de E1 :

A – 3 Sto A
A + 0.03 → StoB
B × B + 1 = …
B + 0.03 → StoA
B Sto A
A × A + 1 = …
A + 0.03 → StoB

210. E2 : Mais il manque encore une étape.
211. E1 : Quoi donc ?
212. E2 : Mais faut savoir répéter ça (il pointe sur deux instructions « A + 0.03 → StoB ; B × B + 1 = … »
213. E1 : Non, tu dois répéter ça aussi (il point sur l'instruction « B Sto A ») car ça permet de remettre de chiffre qu'on a mis dans B.
214. E2 : Il faut savoir faire A plus 0.03 S T O au B et faire B fois B plus 1 et ensuite retransformer B…
215. E1 : En A.
216. E2 : En A, et ensuite A plus 0.03 dans B. Alors le groupe de touches à répéter. (il commence à écrire dans la fiche).
217. E1 : Ensuite, on met – 3 dans A comme départ et on fait …
218. E2 : …B Sto A et ensuite on repart mais je vais pas le mettre car comme ça on va l'embrouiller.
Leur groupe de touches à répéter écrit dans la fiche :

- A + 0.03 → Sto B
- B × B + 1 =
- B STO A

219. E1 : C'est à répéter, 5 divisé par…
220. E2 : 166 fois. Je vais mettre le nom ici.
221. Ob. Vous avez dit, il faut répéter combien ?
222. E2 : Il faut marquer ici ?
223. Ob. Non, non, je vous demande parce que…
224. E2 : 166 fois.
225. Ob. Pour quoi ?
226. E2 : Parce qu'on a fait, qu'est-ce qu'on a déjà fait ?
227. Ppu : Comme la dernière fois, on vous demande surtout de l'écrire, que ça soi écrit, le groupe de touche à répéter.
228. E1 : Par ce qu'on est parti de – 3 sur l'intervalle.
229. E2 : Ah oui, on a divisé 5 par 0.03.
230. Ob. Mais ça donne pas exactement 166.
231. E2 : Oui ça donne 166 virgule 666.
232. Ob. : Mais pour quoi vous prenez, vous avez pris 166 au lieu.
233. E2 : On a pris la valeur…
234. E1 : Petite.
235. E2 : Entière parce que si on prend au-dessus on va sortir de l'intervalle.
236. Ob. D'accord.
237. E (du groupe à côté) : Vous l'avez déjà fait ?
238. E1 : Oui, on a l'automatisme.
239. E : Le A, c'est, c'est moins 3 ?
240. E2 : (il dit à cet élève en pointant sur ce qu'il vient d'écrire sur la fiche) Au début vous parts avec la valeur – 3, ça fait A plus 0.03 S T O au B et tu as cette valeur et ensuite tu fais le B dans A tu repart de A etc.
241. E : On a fait un truc avec Ans mais chaque fois que tu mets égale ça change (il rit).
242. E2 : Ca marche pas, quoi. On essaie. (il écrit dans leur brouillon) :
3 + 0.03
Ans + Ans + 1 = …

243. E2 : Ca, parce que – 3 //
244. Ppu : Vous écrivez votre groupe de touche à répéter. Mettez bien dans le nom du groupe pour que je puisse bien interroger.
245. E1 (à E) : En fait on a pas utiliser – 3.
246. E2 : Ouais c'est le départ, après t'as pas besoin de répéter.
247. E1 : Si on utilise ça il faut commencer à – 3 plus 0.03. ca fait, ça fait…ce qui fait…
248. E2 : Ca marche.
249. Ob. : Ca marche avec quoi, avec Ans ?
250. E2 : Non, on a fait comme ça. J'ai vérifié pour voir si ça marche.
251. Ob. : Pourquoi vous avez pas vérifié avec Alpro?
252. E2 : Parce qu'on a que des lettres ? Parce que – 3 au début…
253. E1 : Il faut commencer avec ça si on veut avoir – 3 aussi.
254. Ppu : C'est bon je peux ramasser ? Ce groupe ?
255. E2 (à E1) : C'est quoi ?
256. E1 : C'est quoi, c'est quoi ça ?
257. E1 (Il écrit dans le brouillon) : C'est moins, ouais, c'est…Parce que ça, ça fait – 3. Ah, non. C'est – 3…
258. E2 : Non, il faut commencer avec – 3.003. Non.
259. E1 : Non, car si tu commences avec un A égale à ça le 1er chiffre c'est – 3 et c'est toujours dans l'intervalle.
260. E2 : Mais tu commences pas avec ça. Tu fais ton truc. Tu fais – 3 fois – 3 plus 1 ça donne ton truc. Et ensuite, tu commences à répéter les touches.
261. E1 : Ah, d'accord.
262. E2 : Mais les touches à répéter ça veut pas dire que c'est que les touches que tu vas utiliser, ce sont touches que t'as besoin de répéter. Ca, t'as pas besoin de répéter. Ca tu fais une seule fois. Et ensuite…
263. E1 : Tu peux partir avec ça, c'est juste A égale à… moins (il écrit A = - 3.003).
264. E2 : Mais je sais pas si on peut prend avec ça.

265. E1 : Pourquoi pas, de toute façon ça va pas calculer avec ça.
266. E2 : Ouais, ouais, tu peux partir de ça, si tu veux.
Sur le brouillon :
A = - 3.003
3 + 0.03
2.97 + 0.03
A = - 3.003
3
3 × - 3 + 1 = …
____
____
____

267. E (du binôme à côté à E2) : Ca c'est Calculator qui doit répéter.
268. E2 : Mais Calculator c'est nous.
269. Ppu : Bon alors ce que je vous propose maintenant c'est de regarder votre production avec l'idée qui est la suivante : tout le monde se fait juge de la possibilité de la réalisation par Calculator du message que vous lui envoyez. D'accord ? Caculator je vous rappelle que qu'il exécute ce qu'on lui a envoyé en répéter. On va imaginer qu'il répète ça et on va regarder si on obtient ce qu'on lui souhaite faire. Je vous rappelle qu'il imprime, imprime automatiquement chaque fois qu'on appuie sur le signe égale. On va en prendre quelques uns. Je ne crois pas tout prendre. Ici je commence par celui qui est le plus court donc ça devrait être plus intéressant. Je vois (il commence à écrire au tableau) Ans fois Ans plus 1.
270. Oui oui, venez là s'il vous plaît.
Au tableau :
Ans × Ans + 1

271. Ppu : Il ne s'agit pas de rire. Il s'agit de, je vous rappelle, que se demander si, si ce qu'on veut que Calculator fasse est bien fait par Calculator en imaginant que ce groupe est répété. Ce groupe est répété, est-ce que Calculator a réussi à faire ce qu'on voudrait qu'il fasse. Votre commentaire, s'il vous plaît. Assez fort, s'il vous plaît. Pour que tout le monde entende bien.
272. E (de la classe qui prend souvent la parole) : La première fois, ça va marcher.
273. Ppu : Oui ça va marcher pour la 1ère fois. J'aimerais pas répéter car vous êtes capable de défendre votre argument…Je répète pas.
274. Elève 1 (un autre) : On sait pas si Ans est le résultat de ça ou pas.
275. Elève 2 : Le Ans, il est stocké.
276. Elève 1 : Car vous savez, quand vous savez que A égale à – 3 et B égale à 0.03 et quand vous faites opération, quand vous faites A plus de quelque chose, de B, A plus B et vous faite égale et là, vous avez Ans. Et dans ce cas-là, Ans c'est x.
277. Ppu : Mais le B ou il est, là ? Je vois le B. Ou il est ?
278. Es : (rient)
279. Elève 1 : En fait il ne maquait que le x. Vous prenez, vous prenez, vous faites - 3 et vous faites – 3 fois un nombre de, de 1 jusqu'à 166. Et là, 166 fois 0.03 et là vous trouvez x. En fait si vous faites sur la calculatrice x c'est, Ans c'est x.
280. Elève 2 : Mais le Ans, il varie chaque fois là.
281. Es : Oui, il varie.
282. Ppu : Oui d'accord, mais comment concrètement Calculator sur Alpro, d'abord je ne vois pas le signe égale. Ou est-ce que je mets ? Je mets là. Ou ici ? Donc on va imaginer qu'Alpro, pardon Calculator imprime ça, après le signe égale. C'est bien ça. Il fait ça. Est ce qu'il va obtenir successivement les valeurs qui nous intéressent ?
Au tableau
Ans × Ans + 1 =

283. Es : Non.
284. Elève 2 : //
285. Elève 1 : Je veux dire qu'il a des choses avant mais ici c'est seulement le Ans, ce qu'il fait à chaque fois.
286. Ppu : Ah, il y a des choses avant ? Je mets A plus B ? Et ?
287. Elève 1 : Oui. Et égale quelques choses et c'est quelques choses, c'est Ans. Oui c'est A plus B mais là on peut pas répéter, c'est seulement ce qu'il répète à chaque fois.
288. Elève 1 : A plus B on peut pas répéter car devant B il y a des choses qui varient de 1 à 166. C'est plus rapide chaque fois que chaque fois il fait A plus B et égale car s'il fait Ans fois Ans plus A, c'est forcement plus long.
289. Ppu : Oui mais on peut pas trancher là ? J'aimerais qu'on tranche, qu'on sache. Car ici vous hésitez. Et ici le Ans, il vient de quel « égale » ?
290. Elève 1 : C'est A + B.
291. Ppu : Alors A plus B égale.
Au tableau :
A + B = || Ans × Ans + 1 =

292. Ppu : On va imaginer comme ça. Et il appuie sur A + B et égale. Et il va donner quoi ?
293. E1 : En fait c'est x. Là, c'est donné x. Ans c'est x.
294. Ppu : Oui d'accord. Admettons qu'il y a une certaine valeur de x ici. Il va faire x fois x égale et il recommence ici. Ce n'est pas la valeur suivante de x car c'est le résultat de x, donc il risque d'y avoir une répétition sur les valeurs de f de x. Donc c'est-à-dire, c'est les valeurs de f de x qui sera mis au carré plus x mais pas les valeurs de x. D'accord ? Je voudrais qu'on est plutôt d'accord pour éliminer ça. Alors j'en prends un autre car je peux pas tout prendre. Alors je vois le groupe de touches à répéter là. (il commence à écrire au tableau) A plus B Sto B et voilà je vois ça comme groupe à répéter.
Au tableau :
A + C Sto B

A × A + 1 =
B × B + 1 =
B + C Sto A

295. Ppu : Voilà, je vois ça, comme groupe à répéter. Chacun le regarde déjà et s'il veut bien juge.
296. E : A c'est – 3.
297. Ppu : Il est vrai que ce qui est précédé de, comme le débat est venu comme tout à l'heure, c'est – 3 Sto A. C'est marqué ici « étape de préparation ». Moins 3 Sto A et 0.03 Sto C. Alors…
Au tableau :
- 3 Sto A A + C Sto B
0.03 Sto C A × A + 1 =
B × B + 1 =
B + C Sto A

298. Ppu : Chacun se prononce ah ? (…) Alors ? Moi, ce que je redis j'essaie de suivre ce qui va, en imaginant que Calculator appuie sur les touches, ça. Alors – 3 Sto A, 0.03 Sto C. Alors A + C Sto B alors dans B il y – 2.97. Et il fait A fois A + plus 1. Ca fait 9 plus 1, 10, c'est 10 c'est-à-dire la première valeur de la fonction. Et B fois B + 1, c'est – 2.97 fois – 2.97 plus 1, c'est la 2e valeur de la fonction. Et B + C Sto A alors, c'est quoi cette étape (il pointe dans la phase « B + C Sto A » au tableau) ? B c'était – 2.97 et C c'était 0.03 et donc c'est – 2.94 dans A. Et c'est écrit dans le groupe à répéter. Donc je reviens, A c'était - 2.94, plus C c'était 0.03 donc c'est – 2.91 que j'ai mis dans B. – 2.94 fois – 2.94 plus 1 et puis – 2.91 fois – 2.91 plus 1, égale B plus C dans A et je recommence. Est-ce que c'est valide ou pas ?
299. Es : Non, non.//
300. E : Il y a une faute de succession.
301. Ppu : Il y a une faute de succession, c'est-à-dire ?
302. E : C'est-à-dire s'il répète tout ça il donne 2e fois de //
303. Ppu : Ah, il y a des répétitions. C'est ça ? Des choses qui vont revenir deux fois ?
304. E : Non, la 1re fois ça met //, et la 2e fois on part dans B // Ah, non, non, je me suis trompé.
305. Ppu : Alors, je voudrais que chacun se prononce dans sa tête. Est-ce que c'est valide comme des groupes à répéter.// Et ma question c'est qu'est-ce qu'on peut faire plus simple avec la même idée ? Alors, je vous propose quelque chose que j'ai l'impression que ça ressemble un petit peu. Je prononce pas encore complètement là-dessus. Voilà (il prend un autre message et l'écrit au tableau), je vois ici, – 3 Sto A, A plus 0.03 Sto B, B fois B plus 1 égale et voilà, je vois ici, à répéter, ce groupe à répéter. Ca paraît plus simple. (et il dessin un contour autour de ce groupe sauf - 3). Ce que je vous demande c'est, est-ce que si Calculator fait ça, est-ce qu'il va bien fournir à l'impression, toutes les images, les valeurs successives ?
Au tableau :
3 Sto A
A + 0.03 Sto B à répéter
B × B = 1

306. E : Est ce que c'est les 3 étapes à répéter ou 2 ?
307. Ppu : Ah, non, non. C'est Christophe et … (à ces élèves). Est-ce que j'ai falsifié ?
308. E (du binôme ayant écrit le message) : Non.
309. Ppu : Bon, j'ai repris mais vous avez le droit de critiquer ah ?
310. E2 : Il faut pas répéter la première.
311. Ppu : Il faut pas répéter la 1ère ? Mais pourquoi ?
312. E2 : C'es la valeur de départ et ensuite…
313. Ppu : C'est la valeur de départ. Alors toi, tu proposeras de répéter seulement ça ?
314. E2 : Ouais.
315. Ppu : Ouais. Est-ce qu'il y a des avis contraires ?
316. Es ://
317. E : Si on met – 3 dans A alors chaque fois qu'on continue ce n'est pas les valeurs successives de x.
318. E : C'est toujours la même image.
319. Ppu : Attends, je prends, on va imaginer, alors – 3 est stocké dans A et puis on ajoute 0.03 dans A,
320. E1, E2 : Non, ça marche pas.
321. Ppu : Donc dans B alors chacun retient dans A il y a – 3 et dans B il y a – 2.97. Puis il fait – 2.97 fois - 2.97 et plus 1 donc il obtient l'image donc de ? De – 2.97. Qui est imprimé. Je ne crois pas qu'il a obtenu l'image de – 3 ?
322. E : Si.
323. E : Non.
324. Ppu : Non. Il n'a pas obtenu l'image de – 3. Mais regardons la suite. Donc, je reviens ici, Sto A, mais qu'est-ce qui est stocké dans A ? Là (il pointe sur B fois B + 1) Car répéter ça veut dire que Calculator arrive ici (il point sur le signe =) et puis il revient là (il ponte sur Sto) Donc qu'est ce qu'est stocké dans A ? Ce qui est là ? Alors, est-ce que ça, ça peut rectifier un peu ? On peut améliorer un peu ?
325. E2 : Il faut ajouter, à la fin on va mettre, B S T O au A.
326. Ppu : Oh, là
327. E : Non, non.
328. E2 : Il faut, faut déjà, il faut partir avec A égale à - 3.03.
329. Ppu : Oui, ca veut dire que supprime ici (il point sur Sto A) et je fais ça ? Je fais ça (il fait une ligne de séparation) ? Et ce qui est à répéter c'est comma ça ?
330. E2 : Non, non, il faut changer.
331. Ppu : Ah, il faut changer ?
Au tableau :
3 Sto A
_____
A + 0.03 Sto B à répéter
B × B = 1

332. Ppu : Mais est-ce que ce groupe à répéter c'est pas mal quand même ? (il point sur la 2e partie après le trait de séparation).
333. E : Oui.

334. Ppu : Oui. Est-ce qu'on retient quand même ? On va retenir ce groupe à répéter. On voit qu'il y a un problème mais on sent que la répétition est assurée, non ? La répétition est assurée. Je vous propose un autre groupe de touche (il écrit au tableau) A + 0.03 Sto B, B fois B plus 1 égale et B Sto A. Voilà, je vous l'entoure.

Au tableau :

A + 0.03 Sto B
B × B + 1 =
B Sto A

335. E1 (ils parlent de leur message) : Copie le truc rouge aussi.
336. E2 : B S T O au A.
337. E1 : Il sait même pas copier.
338. E (du binôme à côté dont le professeur vient d'écrire le message au tableau) : Non car c'est pour nous on change A à chaque fois.
339. E1 : C'est ça qui a changer A. C'est la 3e valeur de x qui change automatiquement A
340. Ppu : Voilà, A plus 0.03 Sto B, B fois B plus 1 égale et B Sto A. Ca va ? Chaque fois qu'on retrouve une nouvelle valeur de B. Alors ? C'est-à-dire quand même qu'il y a quelque chose qui s'est passé avant ah. Ouais. C'est-à-dire – 3 Sto A.
341. E2 : Non, - 3.003 Sto A.
342. Ppu : - 3.003 Sto A. Bon on va imaginer qu'on démarre, on est…on ajoute 0.03 dans B et on fait B fois B plus 1, alors on obtient bien l'image par f du nom qui était dans B. Et B Sto A, B Sto A c'est-à-dire que ce qui était là passe dans A et on recommence avec 0.03. Ca paraît, ça peut tenir bien la route cette affaires comme groupe à répéter là. On voit qu'il y a des petits peu de problème mais on sent qu'il y a une répétition qui est possible là. C'est possible là, non. Ca se ressemble là. Non ,
343. E2 : Il ne manque qu'une étape.
344. Ppu : Une manque une étape, ouais. Ouais. On voit qu'il y a plusieurs possibilités. Je vous propose deux possibilités qui se dégagent de tout ça. (il commence à écrire au tableau). Si vous voulez, j'efface. Voilà, je vous propose une première idée. Je vous donne le premier groupe de touches. J'essaie de rassembler un peu. A fois A plus 1 et A plus 0.03 Sto A. Qui sert là des groupes à répéter. Ouais ? Il y a certainement des choses à faire mais il y a un groupe à répéter. On a vu 2e idée qui est apparue. A fois A plus 1, A plus B fois 0.03 Sto A et B plus 1 Sto B. Tout à l'heure on a vu. 0.03 Sto A et B + 1 Sto B. Voilà des candidats, je les rappellerai des candidats à être répétés.

Au tableau :

A × A + 1 =	A × A + 1 =
A + 0.003 Sto A	A + B × 0.03 Sto A
B + 1 Sto B	

345. Ppu : Mais on a vu que pour assurer la répétition, il fallait forcément prévoir quelques choses, parce que là nous sommes dans l'idée que ça c'est les groupes à répéter. Mais il va falloir dire ça à Calculator là. Il faut répéter ça. Il fallait lui dire. Cela nous amène à la question suivante : comment faut-il dire à Calculator il faut répéter ces groupes ? L'un de ces groupes, enfin. On va vous proposer de le faire et d'avoir une réaction là-dessus. Je vous donne les fiches, chaque binôme par fiche. Autrement dit, votre production sera écrite. Vous prenez note l'un de deux groupes. Vous allez remarquer l'un ou l'autre. Je vous demande d'utiliser l'un ou l'autre et de réfléchir comment faut-il dire ça à Calculator. Et comment on pourrait dire ça, je vous propose de le dire avec des mots, des mots de la langue française.
346. E1 : Il a // vicieux.
347. Ppu : Avec des mots de la langue française. Alors nous sommes d'accord que Calculator est capable de répéter l'exécution des groupes de touches que je vous ai écrits au tableau, qui sont au tableau, vous choisirez ce que vous voudrez. Et on va imaginer que Calculator peut comprendre 5 mots de la langue française qui aide à assurer la répétition, à contrôler la répétition le groupe affiché. Alors ce qu'on vous demande c'est de choisir les mots. Attention de ne pas en choisir trop car on vous demande au maximal 5 mots. Et d'écrire carrément le message à Calculator avec les mots, y compris les mots. Donc vous choisissez l'un des groupe et vous mettez des mots, l'écrire avec des mots. D'accord. Mais attention que vous avez seulement 5 mots.
348. E2 : 5 mots ?
349. Ppu : Et l'idée est la suivante, du coup, grâce à ces 5 mots choisis, vous allez écrire un message à Calculator qui devrait nettement beaucoup plus court qu'à ce que vous avez calculé la dernière fois. Puisque la dernière fois, certains d'entre vous étaient arrivés à quatre milles, quatre milles. Avec cela, vous devez descendre largement au-dessous. Pour cela vous allez compter des mots comme des touches. Chaque fois que vous mettez des mots dans le message, vous avez droit au plus de 5 mots, chaque fois que vous mettez des mots dans le message vous comptez comme une touche. Et vous devriez avoir des mots et des touches qui sont nettement inférieurs qu'à ce que vous avez la dernière fois.
350. E1 : Au boulot.
351. E2 : Il faut trouver des mots à utiliser ? A répéter ?
352. E1 : Répéter 166 fois.
353. E2 : Ca fait 3 mots ça ? Et ensuite…
354. Ob. Non c'est 2 mots. Il y a en fait répéter et fois.
355. E2 : Ah, ça compte pas les chiffres ? Donc répéter, mais les lettres ça compte pas comme des mots ? A ça compte pas ?
356. E1 : Non, car c'est d'Alpro.
357. E2 : Donc tout ce qui se trouve là (il point Alpro) ne compte pas ?
358. Ob. Non, tout ça, ce sont des touches donc on compte comme des touches. A c'est une touche etc.

359. E2 : D'accord. Ca sera donc répéter 166 fois. A plus A, A fois A plus 1.
360. E1 : A plus 0.03 S T O au A et A fois A plus…
361. E2 : Ca c'est quoi alors (il commence à écrire le message).
362. E1 : Mais je crois pas que c'est parfait.
363. E2 (il écrit au brouillon) : Répéter 166 fois…Il faut mettre, en partant de …Répéter 166 fois, A fois A plus 1, A plus 0.03 S T O au A. En partant de, 1, 2, 3, 4 (il compte les mots dans le message). Non.
364. E (du binôme à côté) : Monsieur, il faut mettre des touches ?
365. Ppr : Il faut faire un message à Calculator qui contient des groupes à répéter qu'on veut et qui lui permet avec des mots et qui lui, en imaginant que si Calculator comprend ces mots, qui lui assure bien ce qu'on lui demande de le faire, imprimer, chaque fois qu'on appuie sur le signe égale, imprimer toutes les images et seulement ces images. Je rappelle que toutes les images et seulement ces images.
366. E1 : Tu mets des // je sais pas mais.
367. E2 : Mais avec les touches que tu peux trouver. Je pensait aussi, « à partir de l'intervalle » mais il y a pas de touches donc…T'es obligé Est ce que ça marche ? Tu pars de - 3et tu fais 166 fois et tu vas arriver à, à 2.
368. E1 : Ca marche, ça marche mais…c'est pas propre, tu vois. Si on doit écrire, combien de fois on répète, c'est pas…il y a des étapes avant où 5 est divisé par 0.03.
369. E2 : Mais ça, c'est ce que tu rentres dans Calculator.
370. E1 : Oui, d'accord. Mais je dis que ça marche.
371. E2 : Car en faisant des conditions, soit, tu trouves pas des touches, soit ça fait trop de mots…On marque ça ?
Au brouillon (ce que E2 écrit) :
Répéter 166 fois A × A + 1 =
A + 0.03 STO
En partant de - 3 STO A

372. E1 : Non, attends un peu. On a tout le temps.
373. E2 : Si tu trouves un autre truc.
374. Ob. : Pourquoi vous écrivez en partant de ? Parce que si vous avez écrit – 3 STO A.
375. E1 : On écrit just – 3 STO A et on enlève ça, en partant de. Et répéter c'est juste ça à répéter (il point sur les 2 premières instructions) ce n'est pas tu ça donc tu mets tu ça au début et ça c'est après.
376. E2 : En fait je vais faire – 3 STO A et répéter…
377. E1 : Voilà.
378. Ob. : Car il faut que votre message soit court aussi.
379. Ppr : On lui reste 5 minutes. Il va y avoir l'un des 2 groupes, d'autres touches et des mots d'accord ?
380. E2 (il écrit dans le brouillon) : Voilà, ça fait 2 mots quoi.
Leur 2e message dans le brouillon :
3 STO A
et répéter 166 fois A × A + 1 =
A + 0.03 STO A

381. Ppu : On dit que qu'il y a 5 mots. Vous prenez ça, l'un ou l'autre, des groupes de touches. Et vous ajoutez des mots et des touches, de telle manière que le message permette à Calculator d'exécuter exactement ce qu'on veut obtenir de lui. C'est-à-dire qu'on veut obtenir de lui, toutes les images de la fonction. C'est ça ce qu'on veut obtenir de lui. Toutes mais seulement ces images…
382. E1 : Il faut que tu fasses avec un feutre.
383. Ob. : Et tu écrits un peu plus grand, pour que les autres voient ça aussi.
384. Ppu : Bien entendu vous pouvez utilisez Alpro ah, parce que Calculator appuie sur Alpro…
Leur final message :
3 STO A
Répéter 166 fois A × A + 1 =
A + 0.03 STO A

385. E2 : 1, 2, …22, 23.
386. E1 : Pourquoi 22, 23.
387. E2 : Pour calculer la longueur du message, les mots sont comptés comme des touches…23, c'est ça.
388. Ppu : Ecrivez votre message, s'il vous plaît. Oubliez votre nom…, s'il vous plaît.
389. E1 : 166 c'est un mot, en fait c'est pas des chiffres ?
390. Ob. : Ca c'est un mot, A par exemple, mais ça c'st comme 1, 2, 3, par exemple.
391. E2 : C'est 23.
392. Ob. : Ouais, vous marquez votre nom, s'il vous plaît. Vous avez déjà fait la programmation ?
393. E1, E2 : Non.
394. Ppu : Je peux pouvoir ramasser ? On va regarder. Bon, je ramasse. Oui, comptez tous les mots…La longueur du message.
395. E (du binôme à côté) : Vous avez mis quoi comme message ?
396. E2 : Ah, ha, Répéter 166 fois.
397. E (à son ami de groupe) : Mets bien – 3 STO A au début. Il faut faire une phrase ?
398. E2 : Avec 5 mots. Il faut lui dire ce qu'il faut faire avec 5 mots au maximum. Autant des lettres et chiffres que tu veux mais 5 mots au maximum, différents.
399. E : T'as 5 mots ?
400. E2 : Tu dois mettre 5 mots au maximum.
401. E : Multiplier, mémoriser.
402. E2 : Mais mémoriser, t'as pas besoin. T'as Sto pour ça.
403. E : Qu'est ce qu'on fait ? Qu'est ce qu'on fait ? A ? Etapes 1, étape 2 ?
404. E2 : Etapes ? Je sais pas.
405. E : Je mets 1, 2 alors.
Le message de ce binôme :
- 3 Sto A 0)
A × A + 1 = 1)
A + 0.03 Sto A 2)

0) 1) 2) 1) etc.

406. Ppu : C'est bon, je peux ramasser ? C'est bon, il me manque encore deux affiches…Bien rapprochez pour ceux qui veulent voir les messages. (Ca sonne). Prenez du temps pour regarder quand même. Comparez le nombre de touches à appuyer. La question qu'on aurait aimé travailler avec vous c'est si Calculator comprend les mots que vous lui donnez, est-ce qu'on réussit à lui faire exécuter ce qu'on souhaite obtenir, c'est-à-dire toutes les images de la fonction et seulement ces images. Je vois, par exemple ceci, « répéter 166 fois » des groupes… l'un des groupes qui étaient donnés, il y a deux mots « répéter » et « fois » et la longueur du massage c'est 20. Est-ce que vous pensez que c'est bon ?
407. E : Ouais
408. E 2 : Non, car on sait pas d'où on part ?
409. Ppu : Ah, on sait pas d'où on part. Donc ça nous allons appeler l'initialisation, la préparation d'une //. Sinon nous avons « A fois A + 1 égale, A plus 0.03 Sto A, afficher les valeurs après égale ». Mais afficher les valeurs après égale, il // de toute façon, ah. On sait pas d'où on démarre. Mais ici, non seulement on sait pas d'où on démarre mais une autre chose c'est autres choses ?
410. E 2 : On sait pas combien de fois on répète.
411. Ppu : On sait pas ou on s'arrête. Si vous lui demandez, Calculator il sait pas quand il s'arrête. Ici je vois… « - 3 Sto A, A + 0.03 Sto A, A fois A +1 égale, A + 0.03 Sto A et refaire avec la nouvelle valeur de A ». Alors, oui, mais ?
412. E : Combien de fois ?
413. Ppu : Oui, combien de fois ? Et bah, il faut savoir qu'il s'arrête car si vous lui dites pas. Et (il lit un autre message) «A fois A plus 1 égale, répéter A plus B fois 0.03 Sto A, remplacer A, B plus 1, remplacer B » alors remarque ?
414. E2 : On sait pas combien de fois il faut faire ?
415. Ppu : On sait pas combien de faire il faut faire. Alors ici « - 3 Sto A, A fois A plus 1 égale, A plus 0.03 Sto A, refaire phase 1 » donc refaire ce groupe, 166 fois. C'est tout bon ? Alors, il y a 3 mots et ça 25. Vous pensez que je me trempe sur le décompte des mots ? Ca c'est plutôt un commentaire pour Calculator. Il y a celui-ci, prenez le temps. Et ce message, répéter fois, répéter 166 fois, c'est pratiquement pareil et il a mis deux mot avec ce qu'on a appelé la condition d'arrêt et l'initialisation. Avec ce message, il y a que 23 touches et deux mots et on peut obtenir exactement ce qu'on lui demande. Voilà écoutez bien, nous sommes très contents de travailler avec vous et j'espère que…
416. Ob. : On peut leur dire qu'ils ont travaillé, ils écrivent des programmes informatiques… à une machine don Calculator est une partie.
417. Ppu : Alors Calculator, vous voyez bien qu'on imagine bien avec le Calculator un ordinateur et vous voyez bien que cet ordinateur est différente d'Alpro. Alpro sert de partie arithmétique et de logique. On a pas rajouté la logique. Et tout ça pour le processeur de l'ordinateur. Donc pour réussir le processeur de l'ordinateur, ou la partie centrale de l'ordinateur il y a besoin, vous auriez dû constater, il y a besoin de mémoires et vous avez besoin d'Alpro qui est capable d'exécuter des calculs et vous avez besoin de contrôler la répétition. Donc vous avez besoin de 3 unités, Alpro qui sert de l'unité de logique, vous avez l'unité de mémoire et vous avez besoin de l'unité de contrôle. Tout ça vous mettez dans le processeur. Vous imaginez que le processeur est capable d'enregistrer des données et de sortir de résultats de retours. Et vous avez tel que vous avez sous les yeux un ordinateur. Mais peut-être avec des images en plus.
418. E : Rient.
419. Ppu : Merci bien en tous cas. On est très heureux de travailler avec vous.

I.2. Binôme de la classe 1er V : de la mémoire Ans à la mémoire effaçable

1. Ppu (en distribuant les feuilles) : N'oubliez pas de marquer votre nom et prénom là-dessus.
2. Es : On va entre exercice 2 ?
3. Ppu (en lisant l'énoncé) : Vous aves 10 minutes, attention pour cette partie, vous avez 10 minutes. Vous devez calculer les images de la fonction $f(x) = x^2 + 1$ avec x appartenant à l'intervalle [- 3 ; 2]. Pour cela, on prend un écart fixe entre deux valeurs successives de x qui est égal à 0.2 et on commence par - 3. Les premiers calculs se présentent ainsi : f de -3 est 10, f de – 2.8 est 8.84, f de – 2.6 est 7.76 . Ca on a déjà fait les calculs pour vous. La 1ère question est quel est l'écart choisi entre deux valeurs successives de x ? La 2e est, calculez les images par la fonction f de la sixième, la onzième et la treizième valeur de x…Vous présentez votre solution dans votre feuille, n'oubliez pas de marquer votre nom et prénom. Je les ramasserai après 10 minutes…S'il vous plaît. Alors dans la 2e partie, vous avez deux choses à faire. La 1ère partie, attention, s'il vous plaît. La 1ère partie, vous travaillez individuellement et la 2e en binôme. C'est-à-dire, ceux qui travaillent avec Alpro, vous le ferez sur Alpro. Les autres, vous travaillez avec votre calculatrice. Vous n'oubliez pas de noter votre nom et la marque de votre calculatrice. Attention à ne pas manquer le nom et prénom. Pour cette partie, vous aurez 10 minutes. Après ces 10 minutes, je ramasse les feuilles et je ferai la correction (l'enregistrement n'est pas fait durant ce temps) (…)
4. E2 : / De – 3 à 2/
5. E1 : Ici, x, l'écart entre les x est 0.2. On doit prend 0.2 multiplié par la distance.
6. E2 : 7.76…(…)
7. Ppu : Il vous reste encore deux minutes. (…) Vous avez terminé. Toute la classe a-t-elle fini ? Alors je vais ramasser les feuilles. Ecrivez votre nom. Vous voyez qu'ici, qu'est ce qu'on vous demande ?

On vous demande deux choses. La première, c'est de calculer l'écart entre deux valeurs consécutives de x. Allez. Attention, ces garçons-là. Quel est l'écart entre deux valeurs de x ? Vous l'avez trouvé combien ?
8. Es : 0.2.
9. Ppu (il écrit en même temps au tableau) : l'écart, je l'ai écrit en abréviation (KC) entre deux valeurs de x est 0.2. Il me semble que toute la classe a trouvé cette valeur. La question b « Calculer les images de la fonction f correspondant à la 6e, la 11e et la 13e valeur de x ». Vous avez calculé la 6e valeur de x, c'est combien ?
10. E2 : 1.
11. E1 : - 2.
12. Es : - 2.
13. Ppu (il a noté au tableau tout en lisant) : la 6e valeur de x est égale à – 2. Et f de – 2 est égale à ?
14. E1 : f de – 2 est égale à 5.
15. Es : 5, 5 monsieur ?
16. Ppu : Alors f de x égale à 5. Et puis la 11e valeur de x, c'est combien ?
17. Es : - 1.
18. E1 = - 1.
19. Ppu : - 1 et f de – 1, c'est combien ?
20. Es : 2.
21. Ppu : Egale à 2. Et la 13e valeur de x ?
22. E1 : - 0.6.
23. E : - 2. C'est – 2. Monsieur, la 6e valeur de x est – 1.8.
24. Es : Non, c'est bien – 2.
25. Es : - 0,6.
26. Ppu : Et f de – 0,6 est égale à combien ?
27. E1 : 1. 36
28. Es : 1,36.

Au tableau

a) Ecart entre deux valeurs de x : 0.2
b) La 6e valeur de x = - 2, f(- 2) = 5

--- 11e ----------x = - 1, f(- 1) = 2
---13e ----------x = - 0.6, f(- 0.6) = 1.36

29. Ppu : Alors ce sont des résultats corrects. Donc pour cette question, on vous a demandé de calculer les images de la fonction d, de valeurs de x dont l'écart est 0.2. Qui a trouvé la 6e valeur de x égale à – 1.8 devrait revoir sa méthode. Alors pour cette question, on vous a demandé de calculer des fonctions dont l'écart entre x est 0.2. Maintenant, pour la 2e question, vous allez travailler en binôme. Vous allez calculer les images de fonction. Vous venez de faire des calculs des images d'une fonction. Vous l'avez compris ? Je vous distribuerai des feuilles. Pour faciliter l'échange donc il faut qu'il y ait un nombre pair de groupes, alors ces deux garçons vont jouer chacun un groupe. Vous pouvez maintenant travailler sur Alpro c'est-à-dire la calculatrice que vous avez sur l'ordinateur, et non pas sur la calculatrice. Alors, n'oubliez pas de marquer votre nom et prénom.
30. E2 : On écrit mon nom et ton prénom ?
31. E1 : Tu racontes n'importe quoi.
32. E2 : Oui, écrit mon nom et ton prénom.
33. E1 (commence à lire l'énoncé) : « On commence à réduire l'écart entre deux valeurs consécutives de x…».
34. Ppu (en lisant l'énoncé) : Ecoutez moi bien, s'il vous plaît. Maintenant, on réduit l'écart entre deux valeurs de x et le prend à 0.03. Attention, toute la classe. Tout à l'heure, on l'a pris à 0.2. En utilisant la calculatrice Alpro, on commence à calculer comme suivant, f de – 3 est 10. Et avec l'écart de 0.03, on doit calculer f de – 2.97 qui est égale à 9.8209. On voit qu'il y a beaucoup de calculs à répéter. On fait alors appel au robot, appelé CALCULATOR. Ce robot sait faire deux actions, la 1ère est qu'il sait appuyer sur les touches de la calculatrice Alpro à condition qu'on lui indique ces touches par écrit sur le papier ; Vous vous rappelez tout à l'heure quand vous écrivez un message à vos camarades ? Et deuxièmement action, il imprime automatiquement ce qui est affiché sur la ligne de résultat de la calculatrice Alpro après l'appui sur la touche « = » ; Donc chaque fois que la touche « = » est enfoncée il enregistre automatiquement ce qui est affiché sur l'écran. Et il ne sait pas faire autre chose, ainsi qu'il ne possède ni papier ni stylo. Donc comme lorsque vous recevez le message du groupe à côté, vous avez seulement le message et rien d'autre. On décide d'écrire un message au robot CALCULATOR en lui indiquant les touches de la calculatrice Alpro sur lesquelles appuyer CALCULATOR, pour que lorsque ce message est exécuté, il doive imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images.
35. E2 : Oh.
36. Ppu : L'écart entre deux valeurs successives de x étant égal à 0.03. La fonction f en question est égale $x^2 + 1$. Maintenant vous aurez 10 minutes pour écrire ce message à CALCULATOR.
37. E2 (en incitant sur le mot tout) : Ecrire un message pour tous les nombres ?
38. Ppu : Vous avez 10 minutes et travaillez en groupe.
39. E1 : De –3 à 2 ?
40. E2 : De – 3 à 2, il y a combien ? De – 3 à 2…
41. E1 : Mais ce n'est pas entier.
42. E2 : De – 3 à 2, la distance est 5, divisé par 0.03…
43. E1 : Oah.
44. E2 : 166 nombres.
45. E1 : On doit tout écrire ?
46. E2 : 167

47. E1 : f de x = x^2 + 1. Les valeurs correspondantes, f de – 3.
48. Ppu : Vous avez des questions sur l'énoncé ?
49. E2 : Quelles sont d'autres caractères qu'on connaît pas encore. Il y RCL.
50. E1 : Qu'est ce que c'est ?
51. E2 (il parle en français) : Répète conclusion. T'as compris ? Par exemple, 12, égale puis AC.
52. E1 : Non, d'abord tu dois faire le Stocker.
53. E2 : Plus 36, égale…
54. E1 : Tu vois, ça marche pas. Il faut d'abord stocker.
55. E2 : Rcl.
56. E1 : Je l'ai déjà dit, il faut d'abord stocker.
57. E2 : Egale,
58. E1 : Tu ne dois plus faire Ans. Appuies sur AC/ON.
59. E2 : Je n'ai rien fait.
60. E1 : Voilà…
61. Ppu : Je vois que certains d'entre vous n'ont pas encore marqué le nom et le prénom allez il faut que vous le faisiez.
62. E2 : Allez, écris notre nom et prénom.
63. E1 : Je comprends pas. Comment on peut que ça sort tous calculs ?
64. E2 (relit l'énoncé) : « Maintenant on veut réduire l'écart...» je ne comprends rien. Il y a plusieurs calculs à répéter. Ca veut dire quoi ? C'est-à-dire, x^2 plus 1, x^2 plus 1 et encore x^2 + 1?
65. E1 : Ouais.
66. E2 : Alors, avec cette calculatrice si simple, comment on peut faire un tel calcul ?
67. E1 : On peut mettre 0.03 en facteur. On commence par – 3 et on n'a qu'à soustraire 0.03, plus 0.03, plus 0.03.
68. E2 : Est ce que, comme ça, chaque fois, ça diminue une fois de 0.03, le tout ?
69. E1 : Ouais, ouais, si y on ajoute ça diminue.
70. E2 : Alors, si on calcule 0.03 x au carré et plus 1, ça donne combien ?
71. E1 : x,
72. E2 : Ouais, c'est ça.
73. E1 : 0.03 x au carré plus 1 c'est combien ? Ah, attends.
74. E2 : Non, je ne fais que des essais. 0.03 au carré + 1. Oui, c'est bon, 1.0009. Alors, maintenant A c'est ça, c'est cette valeur.
75. E1 : Non, ça ne va pas. Ce x doit diminuer alors…
76. E2 : C'est combien. Tu peux lire ton résultat ?
77. E1 : 9.8209.
78. E2 : Plus, plus A. Non ça marche pas.
79. E1 : Attends, maintenant on va voir de combien ça diminue chaque fois, pour chaque fois ?
80. E2 : x au carré plus 1. x plus 0.03 plus 1. Oh, mon dieu.
81. E1 : On ajoute 0.03, puis 0.03…
82. E2 : Oui, plus 0.03.
83. E1 : Alors comment on fait pour que lui il écrive tous les résultats ? Voilà, - 3 au carré c'est 9 n'est ce pas ?
84. E2 : C'est-à-dire la valeur prédécesseur augmente toujours par rapport à la valeur successeur de 0.03 x plus ceci. Et on a déjà ceci.
85. E1 : C'est-à-dire, x^2 moins
86. E2 : Stocké, ah, voilà, voilà, on peut le calculer, chaque fois…La précédant est la suivante plus…Mais on sait pas.
87. E1 : Ecoute, ce x, ce x^2 diminue, après que x diminue 0.03 (elle fait les calculs au brouillon). C'est diminué. On soustrait 0.03 aussi. On sait pas, normalement, comment on fait. Ceci est égale 0.06…Ca on a déjà pris sur les progressions. Chaque fois qu'on diminue, ça donne n multiplié par 0.03, multiplié 0.03. Par exemple, si on double 0.03. n varie de
88. Ppu : Il vous reste 5 minutes. Il faut que vous écriviez un message de telle sorte qu'on puisse procéder des échanges.
89. E1 : Un message et on peut sortir toutes les valeurs ? Il suffit un calcul et on peut avoir toutes les valeurs ?
90. E2 : Ouais…Ou on cherche la valeur maximale et la valeur minimale. En tout cas, on peut pas trouver toutes les valeurs, 166 valeurs. De toute façon, on peut pas écrire toutes ces 166 valeurs, n'est ce pas. Maintenant, il suffit de trouver la valeur maximale et la valeur minimale.
91. E1 : Mais ça sert à quoi de trouver la valeur minimale et maximale ? Car on nous oblige de trouver toutes les valeurs de la fonction. Cette fonction a pour graphique une parabole, n'est-ce pas ? Cette fonction, n'est ce pas ? On a trouvé quelle formule ?
92. E2 : Parabole, ça diminue, ça diminue, ça augmente jamais.
93. E1 : Ouais, c'est ça.
94. Ppu : Il vous reste 3 minutes. Attention, s'il vous plaît. Vous avez encore 3 minutes pour écrire le message, puis je les ramasserai et on s'y arrête pour aujourd'hui.
95. E2 : Oui, mais on sait pas comment faire.
96. E1 : On comprend pas. 10 diminue…
97. E2 : Mais cette calculatrice n'a que très peu de fonction.
98. E1 : Tu racontes n'importe quoi…On essaie d'écrire quelques choses. Une quelconque instruction. Par exemple, - 3 moins 0.03, le tout met au carrée.
99. E2 : Doucement, je note, - 3 moins 0.03.
100. E1 : Ou on écrit – 3 moins 0.03, moins 0.03 mais si on continue comme ça, quand est ce qu'on peut finir d'écrire toutes les valeurs ?

101. E2 : la valeur minimale de f est 10. Non, la valeur maximale de f est 10. On peut pas écrire toutes les valeurs car c'est une parabole, à un moment donné, elle augmente on ne peut pas écrire en une formule.
102. E1 : Non, on peut tout écrie car il y a une limite à ce côté-là. Bien sûr, il y a plusieurs valeurs, bien sûr. Par exemple, on a trouvé 106 valeurs. Ou bien, on fait suivant la manière suivante. On calcule f(-3) est 10 et f(2) c'est combien ?
103. E2 : f(2) c'est…
104. E1 : f(2) est égale à 5.
105. Ppu : Est ce que toute la classe a commencé le message ? Pour ceux qui n'ont pas encore il faudrait que vous commenciez car il vous reste pas beaucoup de temps.
106. E1 : Mais monsieur, on sait pas quoi écrire.
107. Ppu : Je vous donne encore 2 minutes.
108. E1 : Le problème c'est que on comprend pas quoi on doit écrire. Attends, ici ce n'est pas les valeurs de f(x) qui sont de même distance entre l'une et l'autre mais c'est les valeurs de x…
109. E2 : On fait quoi alors ? On répète ?
110. E1 : - 3 Ans et encore moins 0.03 et Ans et moins 0.03 Ans, c'est ça ?
111. E2 : Le problème c'est qu'on n'a pas trouvé une loi et ce n'est pas parce qu'on ne peut pas appuyer sur les touches.
112. E1 : Une loi c'est comme faire. Et moi, je n'ai même pas encore compris ce que c'est « écrire toutes les images… ». On appuie sur la touche « = » et ça sort les valeurs, c'est ça ? Ou on appuie successivement sur la touche « = » et la calculatrice donne les valeurs ? On essaie quand même, - 3 et A.
113. E2 : Comment on change la valeur avec – 3.
114. E1 : Non, ça ne va pas.
115. E2 : Je donne la langue au chat.
116. E1 : Non, ça va pas.
117. E2 : Il n'y a aucune loi ici, sœur.
118. E1 : Je veux vraiment savoir que chaque fois que x diminue 0.03, de combien f(x) diminue. Si c'est ainsi, on peut faire des calculs. Mais c'est dommage que ce n'est pas ça. Sinon…
119. E2 : Oui, voilà, voilà, tu écris…
120. Ppu : Je vais ramasser les feuilles. Vous notez votre nom et prénom.
121. E2 : - 3 Sto A, c'est ça.
122. Ppu : Je vais ramasser aussi les brouillons et l'après midi du lundi, vous reviendrez ici.
123. E1 : Bon je vais y réfléchir à la maison.
Leur message :

~~3 AC/ON~~
~~Ans – 0.03 × Ans – 0.03~~

124. Ppu : Vous commencerez à 1h30 de l'après-midi du lundi. Et il faudrait que vous soyez dans la même place qu'aujourd'hui. Maintenant je ramasse les feuilles. Donnent moi aussi les brouillons.
Cette séance s'arrête ici. Voici la suite qui se passe le 10/01/05.

125. Ppu : L'autre jour vous avez fait les deux premières question du problème. Je vais donc corriger la 2e question concernant l'écriture des messages. Vous vous rappelez du message écrit lors de la dernière fois ? Je vous distribue alors à chaque binôme deux feuilles sur lesquelles les messages sont recopiés. Les messages de 1 à 4 sont dans la 1ère feuille et ceux qui restent dans la 2e feuille. (Il fait circuler les feuilles).
126. Es : Les feuilles sont-elles différentes.
127. Ppu : Oui, oui. 2 feuilles pour chaque groupe.
Les messages ramassés lors de la dernière séance :

1) Appuyer 0.03 Shift, Sto → A
AC/On. Appuyer – 2.97 *+ A appuyer = Ans × Ans + 1* = 9.6434
Les étapes suivantes sont pareilles. Appuyer AC/On puis recommences.

2) ~~10 – 9.8209 → A~~
128. $f(x_2) = (x_1 - 0.03)^2 + 1$ avec : $x_1 > x_2$.
$x_2 = x_1 - 0.03$.

3) 0.03 → Sto → A
129. $x_1 = -3 \rightarrow x_2 = -3 + A \rightarrow x_3 = -3 + 2A \rightarrow$ … $x_n = -3 + (n-1)A$
$f(x_n) = [(n-1)A - 3]^2 + 1$
$= (n-1)^2A^2 - 6(n-1)A + 10$

4) - 3 + 0.03 Sto A
A × A + 1 =
AC A + 0.03 Sto B
B × B + 1 =
AC B + 0.03 Sto C
C × C + 1 =
AC C + 0.03 Sto A
A × A + 1 = …

5) ~~Sto~~
~~0.03 Sto A~~
~~⇔ 3 + A = Ans~~

6) 0.03 Sto A
~~3 + A = Sto B~~
3 + A × A + 1 =
AC A + 0.03 = B
B × B + 1 =
AC …

7) - 3 Sto → A AC/On
A × A + 1 = AC/ON
A + 0.03 Sto → A. AC/ON
A × A + 1 = AC/ON
Répéter 164 fois

8) 0.03 = Ans – 3 + Ans

130. Ppu : Est-ce que tout le groupe en a reçu 2 ?
131. Es : Oui.
132. Ppu : Vous le voyez, dans la 1ère feuille ce sont des messages de 1 à 4 et dans la 2e c'est de 5 à 8. Ce sont des messages que vous avez rédigés lors de la dernière fois. Je vous rappelle l'énoncé. Vous devez écrire un message à un robot qui s'appelle Calculator. Il ne sait faire que deux choses : la première chose…(il cherche l'énoncé) Je vous lis la consigne « Appuyer sur les touches d'Alpro à condition que ces touches lui sont indiquées par écrit, imprimer automatiquement ce qui affiché sur la ligne de résultat après avoir appuyé sur la touche « = » » Voilà les deux choses qu'il peut faire. Maintenant, voyons le message 1 (Il relit le message et l'écrit au tableau) « Appuyer 0.03 Shift, Sto A. » Est-ce que Calculator comprend cette phrase ?
133. Es : Oui.
134. Ppu : Vous voyez que ceci il comprend pas. Le mot « appuyer ». car il peut pas lire ce mot. De même, la virgule, il ne comprend pas non plus. Il suffit d'écrire « 0.03 Shift Sto A » car vous savez que Calculator ne peut que lire les touches d'Alpro.
Au tableau :
~~Appuyer~~ 0.03 Shift ~~,~~ Sto

135. Ppu : Ou la 2e phrase « les étapes suivantes sont pareilles ». Est-ce qu'il peut le comprendre ?
136. Es : Non. C'est la même chose avec la phrase «puis recommences ». Voyons le 2e message « $f(x_2) = (x_1 - 0.03)^2$ ». Il ne peut pas comprendre non plus les lettres f, les parenthèses, x_1, x_2 et le carré aussi car ça n'existe pas sur Alpro. En brève, jusqu'à maintenant vous ne pouvez écrire que des messages contenant les touches d'Alpro.
Au tableau :
~~Appuyer~~ 0.03 Shift ~~,~~ Sto
$f(x_2) = (x_1 - 0.03)^2$

137. Ppu : En ce qui concerne le message 4. La 1ère ligne « - 3 + 0.03 Sto A », il peut la comprendre ?
138. Es : Oui.
139. Ppu : C'est dans le message 4. « A × A + 1 = » ça il peut le faire. De même «AC A + 0.03 Sto B B × B + 1 = AC B + 0.03 Sto C C × C + 1 = » ça peut être compris par Alpro. C'est pareil pour « AC C + 0.03 Sto A ». Mais en ce qui concerne cette ligne «A × A + 1 = … » (il note au tableau cette phrase). Ce qu'il ne comprend pas c'est quoi ?
140. Es : Les trois points.
141. Ppu : Oui, c'est ça, ce sont ces trois point (il marque un cercle autour de ces 3 points).
Au tableau :
A × A + 1 = **…**

142. Ppu : C'est-à-dire…Qu'est-ce qu'ils veulent entendre par là ?
143. Es : Continuer
144. Es : Etc.,
145. Es : Continuer ou faire jusqu'à sortir le résultat.
146. Es : Répéter 164 fois. 164 fois à répéter.
147. Ppu (à l'élève qui dit la répétition) : On répète combien de fois ?
148. E : Répéter 164 fois.
149. Ppu (Il note au tableau) : Donc les trois points c'est-à-dire 164 fois. Mais vous le savez, Calculator avec ses capacités à présent, il ne peut pas encore comprendre ça. Donc on doit lui tout écrire.
Au tableau :
164 fois

150. Ppu : Maintenant, passons au message 7. Intéressons nous à la dernière ligne du message « Répéter 164 fois » (et il note en même temps cette phrase au tableau). Il peut comprendre ça le mot « répéter » ?
Au tableau :
Répéter 164 fois

151. Es : Non.
152. Ppu : Non. Comme vous le voyez, Calculator ne sait faire que 2 choses, premièrement appuyer les touches d'Alpro et deuxièmement imprimer. Dans cette phrase, nous avons vu aussi que vu les capacités d'Alpro, il faut écrire tous les calculs et c'est très long à écrire. On va passer à la phase suivante. Je vous distribue donc les nouvelles feuilles. Vous gardez les deux premières feuilles pour la suite. Lorsque vous ouvrir Alpro, il faut entrer les mêmes noms et prénom du groupe. Dans cette phase, vous allez travailler pendant 15 minutes. Vous voyez que dans la dernière fois, la longueur du message écrit à Alpro est égale au nombre d'appui de touches de Calculator sur Alpro. Donc dans cette phase, on vous demande de calculer le nombre d'appui de touches au total et on vous demande de l'expliquer. Vous travaillez en groupe et je vais enregistrer ce même binôme. Vous calculer le nombre d'appui de touche que doit faire Calculator.
153. Es : C'est le message de la dernière fois, monsieur ?
154. Ppu : Oui, c'est celui-ci.
155. E2 : Que doit-on faire ?
156. Ppu : Vous travaillez pour cette phase pendant 15 minutes.
157. E1 (elle relit l'énoncé) : « la longueur du message à Calculator est égale au nombre d'appuis de Calculator sur Alpro ». Dans la dernière phase ? On doit regarder cette feuille (les deux premières feuilles que l'enseignant leur a laissé). //
158. E2 : Dans cette phase 3, regarde, on peut utiliser du papier et la calculatrice. La fiche 2 du binôme ?

Ecrit ici. Mais c'est un nouvel exercice, non ? C'est la 3e phase.
159. E1 : Je sais mais il s'agit de quel message ?
160. Ppu : Je vais vous donner une feuille de brouillon. Vous le gardez jusqu'à la fin.
161. E1 : Monsieur, le message ici c'est quel message ?
162. Ppu : C'est le message que vous avez fait lors de la dernière fois.
163. E1 : C'est le message que vous avez corrigé ?
164. Ppu : C'est le message fait par vous la dernière fois. Vous voyez dans ces messages, il y a parmi vous des gens qui ont utilisé, par exemple les points de suspension.
165. E2 : On va essayer d'écrire quelque chose. On commence par – 3, c'est ça ?
166. E1 : Et AC/On.
167. E2 : On commence par – 3. Oui et tu lis, vas-y.
168. E1 : Plus 0.03. Tu vois ça fait déjà combien de touches ?
169. E2 : Doucement, on va compter après.
170. E1 : Non, je compte maintenant. Shift Sto A et plus 0.03.
171. E2 : Vas-y. Tu lis maintenant.
172. E1 : Shift Sto A. Tu dois prendre ce signe-là. Ce signe moins là.
173. E2 : Mais ce signe moins ne peut pas être au début de la phrase.
174. E1 : Alors quel signe de moins. Il faut que tu prennes ce signe. Voilà. Moins 3. On a déjà appuyé sur Ans. Moins 3, tu essaies encore une fois.
175. E2 : 0.03 Sto A.
176. E1 : Voilà, combien de touches déjà ?
177. E2 : Doucement, attends un peu.
178. E1 : Non tu comptes maintenant, on verra la suite.
179. E2 : Un, deux, trois…Voilà 10 touches. Cette flèche est comptée pour 2 touches.
180. E1 : 10 touches ? D'accord. On continue. A × A + 1 =. Oui A × A + 1 =.
181. E2 : On continue, on ne l'efface pas ?
182. E1 : Oui, tu l'efface…= et puis la touche AC/on
183. E2 : 1, 2…Voilà 6 touches.
184. E1 : 6 touches ?
185. E2 : Non 7 touches. Avec la touche « = » ça fait 7.
186. E1 : Voilà, 17 touches, multiplié par 164 fois. Car la suite c'est la même chose. Oui, tu peux faire le calcul : 17 × 164. Ca fait combien ?
187. E2 : Pour quoi 164 ? Je pense que c'est 166 fois.
188. E1 : Oui 164 fois pour trouver tous les résultats. Oui, tu peux refaire la division pour voir.
189. E2 : 17 multiplié par 164 donne 2788. Donc 2788 appuis de touches.
190. Ppu : Il vous retse 5 minutes…
191. E2 : Oui, tu refais le résultat précédent, tu te souviens pas de ce que t'as déjà dicté ?
192. E1 : Je vais écrire en français aussi ?

Leur réponse :

Nombre de fois 2788
Explication : J'ai calculé le nombre d'appuis pour trouver la valeur $f(x_1)$ correspondant à x_1. Puis on multiplie ce nombre par 164 car j'ai trouvé qu'il y a 164 valeurs comme ça et pour chaque valeur on répète le nombre d'appuis comme avec x_1.
Résumé :
Chaque étape a besoin : 17 touches
164 étapes ont besoin : 17 × 164 = 2788 (fois)

193. E2 : Ca y est, on a terminé. Mais pourquoi tu l'as écrit en vietnamien ? Et puis te l'écriras aussi en français. (il lit ce qu'écrit E1). Oh, mon dieu. C'est trop long comme ça. Tu devrais écrire : chaque étape a besoin 17 touches donc 166 étapes ont besoin etc.
194. E1 : Non, je vais résumer.
195. E2 : Egale 2788 fois.(il demande au binôme à côté) Vous avez trouvé combien ?
196. E1 (au binôme à côté) : 164 fois n'est ce pas ?
197. E2 : On compare ces nombres.
198. E1 : C'est combien le produit de 17 et de 164, c'est 2788 ?
199. E2 : Sorry I dont understand.
200. E1 : Pourquoi 10 ? on recalcule vas-y. 1, 2, 3
201. Ppu : Il vous reste 2 minutes. Pour ceux qui ont déjà terminé, vous écrivez votre nom et prénom.
202. E1 : Oui, c'est bien ça. 10…
203. Ppu : Allez toute la classe, vous vous arrêtez. Vous me donnerez votre feuille et aussi les deux feuilles de copies que je vous ai données tout à l'heure. Vous gardez le brouillon…(Et il ramasse les feuilles).
204. E1 : Ca sert à quoi de faire ça ? C'est pour voir comment exécuter ?
205. Ppu : Attention, s'il vous plaît. Dans cette phase, vous avez trouvé environ combien le nombre d'appui de touches ?
206. E : 2400.
207. Ppu : Et vous vous avez trouvé combien ?
208. E1 et E2 : 2700, monsieur.
209. Ppu : Bon 2700. Vous voyez que le nombre d'appuis de touche est énorme et égale à la longueur du message à écrire. Alors maintenant si vous voulez réduire la longueur du message, que proposez vous à Calculator ? Et vous les écrirez ici (il coupe les feuilles pour les distribuer). Par exemple si vous voulez que vous écriviez moins, que proposerez-vous ? Vous avez 10 minutes pour cette phase.
210. E2 : Alors qu'est ce qu'on veut demander ? Une nouvelle touche par exemple.
211. E1 : Par exemple si on veut répéter ce programme. Si par exemple on répète ceci. Et on peut changer les valeurs.
212. E2 : Qu'est ce que tu veux changer ?

213. E1 : C'est-à-dire, la première valeur c'est 0.03 et on l'enregistre dans A et on prend la valeur A. Ca va comme ça, non ?
214. Ppu : Je vous rappelle que vous devez calculer toutes les valeurs de la fonction $x^2 + 1$ avec x appartenant à l'intervalle [- 3 ; 2] dont la 1ère valeur de x est – 3. L'écart entre deux valeurs de x est 0.03.
215. E2 : Au fait, on n'a pas compris l'essentiel.
216. Ppu : Vous avez 10 minutes pour cette phase.
217. E2 : Je pense que ce qu'on a fait c'est correct.
218. E1 : Oui, mais le problème est qu'on ne peut pas tout écrire. Je pense donc qu'il faut écrire // multiplie par le nombre de fois//
219. E2 : Qu'est ce qu'on a déjà fait ?
220. E1 : - 0.3.
221. E2 : Non c'est – 3.
222. E1 : Oui c'est ça. – 3, plus 0.03.
223. E2 : Et quoi encore ?
224. E1 : Shift Sto.
225. E2 : Pourquoi on doit mémoriser ce résultat ? Ca sert à quoi ?
226. E1 : Shift Sto A. C'est pour faire A × A + 1.
227. E2 : Ah, c'est la valeur qui la plus petite n'est ce pas ?
228. El : Et puis après ça recommence. On remplace ça dans cette équation. On remplace A par B puis par A encore.
229. E2 : Donc c'est A × A + 1. Donc le résultat trouvé multiplié par 2 et plus 1.
230. E1 : C'est $x^2 + 1$. Donc ici on calcule la 1ère valeur. C'est – 3 et…
231. E2 : Oh, oh, yes, yes.
232. E1 : Oui, tu fais shift Sto A.
233. E2 : Oui, c'est déjà fait. Et maintenant c'est A × A + 1.
234. E1 : Et Shift Sto A, oui, shift sto A encore.
235. E2 : Non AC d'abord.
236. E1 : Non Shift Sto d'abord. T'es fou. Vas-y refais ça.
237. E2 : Oui, alors c'était comment ?
238. E1 : - 3 + 0.03 Shift Sto A et A × A + 1. AC/On.
239. E2 : Tu devrais me dicter tout, d'une manière complète.
240. E1 : A × A + 1, shift Sto A encore. Maintenant on va voir A est égale à quoi ? Oui tu effaces tout. Pour voir à quoi ça égale. Oui efface tout. Non, fait plus 1. C'est combien ?
241. E2 : 9.8.
242. E1 : C'est bon. Maintenant tu fais Shift Sto cette valeur.
243. E2 : Non, Ca peut pas être comme ça car c'est trop grand.
244. E1 : Non, d'après moi je veux que tu enregistres deux valeurs pour comparer la valeur de A de cette fois ci et la valeur A de la dernière fois. Si c'est un nouveau A on peut poser A //
245. E2 : C'est un nouveau A.
246. E1 : C'est un nouveau A. Alors c'est différent.
247. E2 : Mais, mais c'est encore ce même A car jusqu'à 164 c'est toujours A.
248. E1 : Alors mais c'est quand même un nouveau A ? Tant mieux alors. On va calculer la 1ère valeur. Après on donne cette valeur à A et puis on répète le processus. On répète le même processus mais on remplace seulement – 3 par A.
249. E2 (lit l'énoncé) : « que proposez vous » Alors qu'est ce que t'en dis ?
250. E1 : Je propose que…
251. E2 : - 3 et quoi encore ? Sto…
252. E1 : Après avoir fait la 1ère étape il traduit le résultat…
253. E2 : Voilà, le 1ère A est – 2.97.
254. E1 : On va enregistrer A. Non je veux dire qu'on calcule seulement x .
255. E2 : Mais ce A change, tu comprends ?
256. E1 : Je veux dire x change…x égale à cela, x égale à ceci. C'est-à-dire.
257. E2 : Voyons 9.8, donc c'est un autre A.
258. E1 : Mais ceci est fixe. Il suffit de changer cela. Cette opération ne change pas.
259. E2 : Pour la suivante, on prend A.
260. E1 : Cette opération ne change pas, c'est seulement ceci qui change pas car il faut toujours lui ajouter 0.03. Alors x_2 est A plus 0.03 et puis on continue de faire Shift Sto A. C'est exact?
261. E2 : Attends, alors ici, comment on continue ?
262. E1 : Ici on continue, ce A, n'est ce pas est A plus 0.03.
263. Ppu : Attention, s'il vous plaît, vous devez lire attentivement la consigne. Ici on vous demande d'écrire des propositions pour améliorer le robot Calculator et non pas Alpro. Je vous relis l'énoncé « Pour améliorer le robot Calculator que proposiez-vous ? «
264. E1 : Puis Shift Sto A et c'est OK comme ça ? Donc ici le A dans A × A est aussi le nouveau A, n'est ce pas ?
265. E2 : Ouis, c'est le même A.
266. E1 : Donc comme ça on répète. Ce A…
267. E2 : Donc si c'est ainsi alors ce n'est pas 17 appuis de touches. Est ce que jusqu'ici c'est terminé pour une fois ?
268. E1 : Oui, c'est terminé pour une fois et ça c'est une autre.
269. E2 : Ces deux choses sont différentes.
270. E1 : Ceci appartient à un processus.
271. E2 : On peut pas le multiplier par 164. Ils sont différents.
272. E1 : Le nombre total de ces deux est 164, tu comprends ? Car ça c'est la 1ère étape et les étapes suivantes sont pareilles.

273. Ppu : Je vous signale qu'à présent, le Caculator ne sait que deux choses : appuyer les touches et imprimer le résultat. Donc que proposez-vous à Calculator pour qu'il puisse vous aider à exécuter vos messages.
274. E2 : Ceci est dans une étape et ceci est lorsqu'on a déjà terminé une étape, n'est ce pas ? – 3 plus A, non.
275. E1 : C'est le A là.
276. E2 : Il manque un signe car ceci a un signe de plus que cela. Le signe moins.
277. E1 : A est le résultat de celui-ci, t'as compris ? Car c'est le x augmente de 0.03.
278. E2 : Je sais.
279. E1 : c'est la 1ère valeur.
280. E2 : Tu ne comprends pas ce que je veux dire. Je veux dire qu'on prend celui-ci et le multiplie par 164 fois. C'est seulement celui-ci, n'est ce pas.
281. E1 : Oui.
282. E2 : Mais dans les étapes suivantes c'est le A plus la valeur précédente donc il y a un signe moins qui n'est pas utilisé.
283. E1 : Ah bon. Oui c'est vrai.
284. E2 : Donc il faut diminuer 106 caractères donc c'est 2, 7…
285. E1 : Non ou soustrait 1 caractère seulement. `
286. E2 : Non on soustrait 106, non, c'est 164.
287. E1 : Mais on n'a pas besoin de ceci.
288. E2 : Donc c'est 16 multiplié par 164. Ca c'est exact.
289. E1 : C'est bon, on va écrire comme ça. Après avoir calculé la 1ère valeur de x, non, la valeur première x_1, on va l'enregistrer dans A. Puis calculer A × A plus 1 et puis on propose Calculator de répéter cette expression en remplaçant x – 0.03 par A.
290. E2 : C'est ça. Et il a une touche pour mémoriser la formule. Enregistrer la formule A × A + 1. Mais lorsqu'on enregistre la formule il ne peut pas faire par lui même.
291. Ppu : Est-ce que vous avez fini ? Je vous rappelle que c'est le robot Calculator qu'il faut améliorer et non pas Alpro.
292. E2 : Mais que veut dire « améliorer » ? Ajouter donc une touche « carrée » comme ça on doit pas écrire A × A.
293. Ppr : Vous avez noté ?
294. E1 : Non, pas encore. Donc je vais simplement écrire que chaque fois qu'on calcule ce A, on remplace dans ceci. Et on garde les autres.
295. E2 : Alors comme je t'ai déjà dit. Il faut avoir une…On garde la même formule. // Alors tu écris, il y aune touche qui enregistre la formule mais on ne sais pas comment appuyer (il rit) et le Calculator ajoute lui même 0.03 pour chaque nouvelle opération. Enfin je veux dire ajouter lui-même 0. 03 à A.
296. E1 : Dans la formule.
297. E2 : Mais quelle formule ? Il faut écrire la formule. C'est A × A + 1.
298. Ppu : Est ce qu'il reste encore des groupes ?
299. E1 : Oui, monsieur.
300. E2 (en lisant leur réponse) : Ce me plaît pas trop. Ce n'est pas très faisable.
301. E1 : Il reste encore beaucoup, en fait, je veux dire le nombre à répéter. Il diminue la moitié.
302. Leur réponse :
303. A × A + 1
304. Il y a une touche pour enregistrer ↗ et il suffit que le robot calcule la valeur de x ^ et le remplace dans la formule précédente. (enregistre dans A)

305. Ppu : Je vais passer en revue des propositions. Par exemple, il y a un binôme qui écrit « Il y a une touche pour enregistrer A × A + 1 et il suffit que le robot calcule la valeur de x enregistre dans A et le remplace dans la formule précédente. » Alors, c'est Alpro que vous améliorez pas Calculator.
306. E1 : Oh, c'est Calculator ? Oh, on s'est trompé.
307. Ppu : Ou un autre binôme « Il y a la capacité de répéter les messages ». Donc c'est la capacité de ?
308. Es : Alpro.
309. Ppu : C'est la capacité de Calculator qui peut répéter les messages. Un autre binôme qui propose d'ajouter « ajouter la touche carré et parenthèses » mais ça c'est de ?
310. Es (ils rient) : Alpro.
311. Ppu : Mais ce groupe a proposé aussi à Calculator qui est « Exécuter les étapes et les répéter automatiquement ». Cela est-il correct ?
312. Es : Oui.
313. Ppu : Oui, ou un autre exemple « Exécuter automatiquement les étapes répétitives ». Vous voyez c'est pour le robot Calculator. Et ce même groupe « la capacité remplacer un nombre dans une opération avant par un autre et comprendre la voix humaine ».
314. E1 : ce qui est impossible.
315. Ppu : Ou un autre groupe « ajouter certaines touches ; utiliser le clavier et non pas la sourire ; mémoriser ce qui est déjà fait »
316. E2 : C'est trop général.
317. Ppu : Ca, vous voyez, est ce qu'on peut mémoriser des choses avec Alpro ?
318. Es : Oui.
319. Ppu : Par quelle touches ?
320. Es : A, B, C.
321. Ppu : Vous avez ainsi vu que les propositions que vous avez faites à Calculator c'est que ce robot doit savoir répéter plusieurs fois un groupe de touches. Alors dans la phase suivante, on essaie d'être plus concret.
322. E2 : Mon dieu, plus concret ! J'en suis incapable.
323. Ppu : On va voir quels sont les groupes de touches à répéter et on va le proposer à Calculator et avec la nouvelle capacité que vous avez lui proposé

qui est de pouvoir répéter les actions, il sera en mesure de calculer les images.
324. E2 : C'est-à-dire qu'il sait répéter.
325. Ppu : La question est donc « Ecrire un groupe de touche à répéter ». Donc quels sont les groupes de touches ?
326. E2 : Shift Sto A multiplier, Shift sto A, A plus 0.03…Oui, c'est ça.
327. E1 : Oh, là, je suis fatigué ça prend pas mal d'énergie. On doit creuser la tête.
328. Ppu : Vous terminez cette phase et on prend une pose après.
329. E2 : Mais non, il y a pas de parenthèse. Monsieur, est-ce qu'on peut utiliser les parenthèses ?
330. Ppu : Tu vois Alpro n'a pas de parenthèse.
331. E1 : On doit répéter le groupe de touche A × A plus 1, n'est-ce pas ?
332. E2 : Mais on doit utiliser les parenthèses. On doit reprendre cette ligne comme un sous titre.
333. E1 : Non si on prend ce groupe pour répéter, ça ne va pas.
334. E2 : Ah, oui, c'est exact. On ne prend que ce qui à répéter. On va commencer à répéter à partir de ça. Sto A.
335. E1 : Sto A oui, donc c'est A × A + 1, n'est ce pas ?
336. E2 : Non, non, on commence à partir de ça. A + 0.03 Sto A et ainsi de suite. Tu vois, c'est pas là qu'on commence. A = 0.03 Sto.//
337. E1 : Non, parce que d'abord c'est A et on commence à répéter. Car tu vois c'est A et on n'a qu'à ajouter 0.03 à A.
338. Ppu : Je vous répète que vous n'écrivez que le groupe de touche à répéter. Car je vois que vous lui écrivez des phrases alors qu'il ne les comprend pas.
339. E1 : Tu vois, A plus 0.03 et on obtient un nouveau A.
340. Ppu : Calculator sais appuyer un groupe de touche, imprimer le résultat après l'appui sur la touche « = » et maintenant, vous lui ajouter maintenant la capacité de répéter un groupe de touche.
341. E1 : On continue de prend le A alors ce sera pas le même A mais un autre A.
342. E2 : Ce A oui, on obtient le résultat et on fait la multiplication.
343. E1 : Comme ça on trouve la valeur f(x). Puis on continue de prendre le A c'est-à-dire ce nouveau A et on lui ajoute 0.03.
344. E2 : Shift Sto A.
345. E1 : Non, ce n'est pas la peine. On enregistre plutôt celui-là. Car ça c'est f(x) on enregistre seulement x. //
346. E2 : Alors comment on continue ? Ajoute 0.03 à A ?
347. E1 : Oui, tu tapes maintenant. Je te le montre. – 3 + 0.03. Shift Sto A. C'est OK ? Puis A × A + 1, vas-y.
348. E2 : Oui et c'est « = », n'est ce pas ?
349. E1 : Oui ça sort le résultat, n'est ce pas ? Et tu fais AC/On. Et maintenant tu prends encore A et tu ajoute 0.03. A c'est ce qu'on vient d'avoir avec Shift Sto faits tout à l'heure. Mais pour quoi c'est 0 ? Bon on recommence. Ce n'est pas la peine de calculer f tu calcules seulement les x.
350. E2 : Attends, A + 0.03 et quoi encore ?
351. E1 : Sto A.
352. Ppu : Vous avez fini ?
353. E1 : Pas encore monsieur, attendez s'il vous plaît. Tu ne prends que ça.
354. E2 : C'est-à-dire on ne fait Shift Sto que ça. N'est ce pas ?
355. E1 : Oui c'est ça, lors qu'on trouve ça, on ajoute encore et on fait Shift Sto la nouvelle valeur…
356. E2 (au binôme à côté) : Le groupe de touche à répéter c'est ce groupe là.
357. E1 : Le groupe de touche à répéter c'est-à-dire il suffit d'appuyer sur une touche et ce groupe de touches sera répété.
358. E2 (au binôme à côté) : C'est-à-dire il suffit que tu écrives quelques lignes et on n'a qu'à répéter ces lignes…
359. Ppu : Vous avez fini ?
360. E1 : Non, pas encore. Il faut que tu écrives aussi la 1ère ligne.
361. E2 : Non, ce n'est pas la peine. Il répète pas cette ligne.
362. E1 : je veux dire que tu fais Shift Sto A et lui il répète (…)
Leur réponse :

Le groupe de touche à répéter :
A + 0.03 shi sto A
A × A + 1

363. Ppu : Vous avez terminé ? (Il ramasse les feuilles)
364. Ppu : Attention, s'il vous plaît. Vous fermez les jeux. Je vais corriger vos réponses pendant environ 10 minutes et puis après. Vous allumez Alpro s'il vous plaît. Ca y est ? Vous regardez au tableau. Je vous réptèe que pour cette question, vous devez écrire dans votre feuille le groupe de touches que Calculator a à répéter. Je note au tableau la réponse du 1er groupe (il commence à écrire au tableau en lisant à haute voix ce qu'il écrit). Je note 1 ce message.
Au tableau :

2) A + 0.03 → A
A × A + 1 = AC/On

365. Ppu : Vous voyez, est-ce qu'il peut exécuter ça ?
366. E1 : Non, dans la 1ère ligne.
367. E2 : Car il n'y a pas de flèche.
368. Es : Oui.
369. Es : Non.

370. E2 : Non, il faut appuyer Shift Sto.
371. Es : Non, seulement Sto.
372. Ppu : Vous voyez, il n'y a pas la flèche sur Alpro. Doit-on remplacer par quelles touches ?
373. Es : Shift Sto.
374. Ppu (il note au tableau) : Shift Sto mais est-ce qu'il faut Shift ? On n'en a pas besoin (il barre le mot Shift). Il suffit juste Sto. Et comme tout à l'heure on a déjà amélioré Calculator avec la capacité de répéter donc il va répéter ce groupe de touches comme le 1er groupe de touches.
375. E : On n'a pas besoin de AC/On.
376. Ppu : Oui est-ce qu'on en a besoin.
377. E : Ce n'est pas vraiment nécessaire.
378. Ce n'est pas la peine (il barre ce mot)
Au tableau :
~~Shift~~ Sto
2) A + 0.03 → A
A × A + 1 = ~~AC/On~~

379. Ppu : Donc je vais faisais comme si j'étais Calculator pour exécuter ce message (puis il commence a taper ce groupe de touches sur Alpro). Vous voyez, A plus 0.003 et…
380. Es : Mais on a pas encore A.
381. Ppu : Mais ici j'ai d'abord simplement tapé ce qui est écrit dans le message.
Ce qu'il a tapé :
A + 0,03 Sto A
Le message renvoyé par Alpro : « Erreur. Mémoire non initialisé »

382. Ppu : Donc qu'est-ce qui manque ici ?
383. Es : A, Il n'y a pas encore A.
384. Ppu : Donc on va prend pour combien la valeur de A ?
385. Es : - 3.
386. E1 : - 3 plus 0.03.
387. Ppu : Donc on va donner - 3 à quoi ?
388. E2 : Mais il nous demande d'écrire des touches à répéter ?
389. E1 : Ca ça n'est pas à répéter.
390. E : Mais c'est la première chose et cette chose-là n'est pas à répéter.
391. Es : Mais cette première étape n'est pas à répéter.
392. Ppu : La 1ère étape est-elle à répéter ?
393. Es : Non.
394. Ppu : Oui, c'est ça. Ce qui est à répéter c'est quoi ? C'est là. (il souligne le groupe de touche à répéter). Voilà vous me regardez. Je vais exécuter ce groupe de touches (Il tape sur l'ordinateur en lisant à haute voix).
Au tableau :
3 Sto A
~~Shift~~ *Sto*
1) A + 0.03 → A ⇐ à répéter
A × A + 1 = ~~*AC/On*~~
Ce qui est tapé :
3 Sto A A + 0,03 Sto A A x A + 1 =

395. Ppu : Ca c'est la valeur de x qui correspond à quelle valeur de x ?
396. E : C'est – 2.97.
397. Ppu : Oui, c'est – 2.97 le tout au carré et y ajoute 1. Et il répète quel groupe ?
398. E : Ce groupe là.
399. Ppu : Donc, vous voyez, on a de petites modifications à faire, mais l'essentiel, c'est comme ça. Ou un groupe a écrit comme suivant « enregistrer une expression, Shift Sto A, Shift Sto B, Shift Sto C, puis enregistrer l'écart 0.03 ». Est - ce que Calculator peut comprendre « enregistrer une expression » ? Comme je vous ai déjà dit, il faut lui indiquer des touches à répéter et non pas la phrase. Ou un autre exemple « Touches de mémoire par exemple A, B, C…, touche de mémoire d'une expression longue, les étapes à appuyer » vous voyez il ne peut pas comprendre. Ou un autre groupe comme ça (il écrit au tableau en lisant à haute voix ce qui est écrit). A plus 0.03 Sto A, A multiplie avec A plus 1. Vous voyez que ces deux-là sont semblables mais il faut bien sûr y ajouter – 3 Sto A. Et on répète ce groupe de touche.
Au tableau :
2) - 3 Sto A
A + 0.03 Sto A
A × A + 1 =

400. Ppu : Il y a deux autres groupes ont donné le même groupe de touches. Un groupe a par contre donné le groupe suivant. A égale – 3 + 0.03n et n égale 1, 2, 3, 4 etc.
401. Es (rient) : On comprend rien.
402. Ppu : Voyons ensemble, est-ce que le robot peut comprendre ?
403. Es : Non.
404. Ppu : Vous voyez il y a « n » qu'il ne comprend pas. Et « n = 1, 2, 3, 4…» c'est les points en suspension qu'il peut pas comprendre bien que vous vouliez exprimer par là une répétition. Donc ici, il y a le 2e groupe de touches qui peut être utilisé (il souligne ce groupe de touche). Maintenant qui peut modifier ça (il indique le dernier groupe de touche écrit au tableau).
Au tableau :
A = - 3 + 0.03n
n = 1, 2, 3, 4…

405. Ppu : Ce groupe s'appuie sur quelle formule pour calculer les images de f ? Non ? Ici il utilise la formule x_n = - 3 plus avec quoi ? – 3 + (n – 1)0.03 (il note cette formule dans un coin au tableau). Par exemple, la 1ère valeur de x, n = 1 donc x est égale – 3

+ (1 – 1)0.03 ce qui est égale à – 3. Une fois trouvé x_n, on calcule $f(x_n)$ suivant la formule $f(x_n) = x_n \times x_n$.

Au tableau :

$x_n = - 3 + (n – 1)0.03$

$f(x_n) = x_n \times x_n$

406. Ppu : Est ce qu'il y a un groupe a utilisé cette formule pour calculer f(x) ? Pour écrire un message à Calculator ? Donc je vais donner un groupe de touche en se basant sur ça. D'abord il faut stocker – 3 dans A et puis on prend la valeur de n dans B. Après ça on calcule A × A plus 1 et puis calculer A + B multiplié par 0.03 et stocker dans A. Ca c'est la valeur de x comme vous voyez dans cette formule-là. Et après ça on augmente le rang de x d'une unité. Donc c'est B + 1 dans B. Et le groupe à répéter c'est le groupe là (il souligne au tableau le groupe à répéter).

Au tableau :

3 Sto A
1 Sto B
A ×A + 1
A + B0.03 Sto A ⇐ répéter
B + 1 Sto B

407. Ppu : Ce groupe de touches est un peu plus compliqué que l'autre précédent. Aucun groupe l'a trouvé…Je vais vous aider à exécuter ce message (il le tape sur son ordinateur). D'abord j'attribue - 3 à A puis 1 à B et je fais A × A plus 1. Ca, vous obtenez 10. C'est la valeur de la fonction en quelle variable de x ?

408. Es : En – 3.

409. Ppu : Je continue A + B multiplié par 0.03 et le l'enregistre dans A. Vous voyez Stocké et ensuite que je fais ? Je prends B plus 1 dans B. Alors maintenant si je veux répéter, je fais quoi ? Quel groupe à répéter ?

410. E : A multiplié par A plus 1.

411. Ppu : Vous avez, c'est la valeur de f en ?

412. E : En – 2.97.

413. Ppu : Et puis je répète comme ça.

Ce qui est tapé :

3 Sto A 1 Sto B A x A + 1 = A + B x 0,03 Sto A B = 1 Sto B A x A + 1 =

414. Ppu : Donc vous avez deux groupes à répéter possibles. La dernière chose que je vous donne à faire. Je vais vous donner à chaque binôme une feuille de format A_3 et ainsi un stylo et tout à l'heure vous me le remettez. Vous écrirez en grand pour que toute la classe puis le voir et on discute ensemble sur ce que vous allez écrire. Vous les faites passer, du papier ainsi que le stylo.

415. E1 : A_3 c'est assez grand.

416. Ppu : Vous marquez votre nom et prénom aussi.

417. E1 : Tu écris trop petit. Personne puisse le voir.

418. Ppu : Vous allez écrire votre message dans cette feuille. Vous écrivez en taille assez grande de manière que vos camarades puissent lire. Vous avez 20 minutes, non pardon, 15 minutes. Attention, vous me suivez, je vais lire l'énoncé (il lit la consigne) « le robot amélioré Calculator II, en plus des capacités de Calculator, doit pourvoir répéter l'exécution d'un groupe de touches ». Ce sont deux groupes écrits au tableau et que je les ai encadrés. «Comment le lui dire ? On suppose que Calculator II peut comprendre au maximum 5 mots de la langue vietnamienne qui aident à contrôler la répétition de l'exécution d'un groupe de touche d'Alpro. Vous devez choisir ces mots et écrire un message le plus court possible à Calculator II. Calculator II, en exécutant ce message, doit imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [- 3 ; 2] et seulement ces images, l'écart entre deux valeurs successives de x étant égal à 0.03. La fonction f est $x^2 + 1$ ».

419. E2 : Pour ce message, que fait-on ?

420. Ppu : Vous avez cet espace pour écrire ce message à Calculator II, ainsi vous avez aussi un espace pour écrire la longueur du message et la liste de mots que vous voulez que Calculator comprenne afin de comprendre la contrôle de la répétition.

421. E : C'est en vietnamien ?

422. Ppu : Oui c'est en vietnamien. Vous avez 15 minutes.

423. E2 : Mon dieu.

424. E1 : Vas-y on commence.

425. E2 : Mais si c'est seulement 5 lettres ce sera impossible. Voyons « lặp » : l, a, p (le mot « lặp » en vietnamien veut dire « répéter »).

426. E (du binôme à côté) : Mais c'est 5 mots.

427. E2 : Non 5 lettres.

428. E1 : Tu rêves ou quoi, c'est 5 mots.

429. E2 :Mais je dis que si c'était 5 lettres ça serait beaucoup plus difficile. Donc c'est tout simplement le mot « répéter ».

430. E1 : Non, c'est répéter continuellement.

431. E2 : Le message.

432. E1 : Oui, on écrit alors ?

433. E2 : Oui tu prends ce stylo.

434. E1 : Le message pour calculer ça. (elle compte des mots)1, 2, 3, 4…

435. E2 : T'as déjà combien ?

436. Ppu : Attention s'il vous plaît. Un mot par exemple « répéter » ça donne 1. Il faut que vous notiez aussi la longueur du message et la liste de mot.

437. E1 : Tu vois, si tu ne mets pas 164 fois, il va faire, faire et il va dépasser 2 // On a déjà trouvé le nombre de fois à répéter mais j'ai oublié la méthode. On va refaire ça, vas-y.

438. E2 : Non, je sais pas.

439. E1 : C'est toi qui l'as trouvé. Alors dis le moi.

440. E2 : Regarde, de 2 à – 3, c'est-à-dire 2 plus 3 ça donne 5. Donc la distance est 5 unité et on divise par 0.03.
441. E1 : Alors ça donne 166 fois.
442. E2 : Mais la plupart a trouvé 164 fois. Donc on le suit.
443. E1 : Oui.
444. Ppu : Vous avez encore 10 minutes.
445. E2 : 10 minutes encore, que fait-on ?
Message :
3 Sto A
répéter ~~continuellement~~ groupe touche 164 fois
A = 0.03 Sto A
A × A + 1 =
Longueur du message : 23 touches
Liste de mots : Répéter ~~Continuellement~~ Groupe touche Fois

446. Ppu : Il faut noter votre nom et prénom. Les messages //
447. E2 (au binôme à côté) : Tiens, on ne nous donne que 5 mots. Donc ce n'est pas la peine d'utiliser AC/On. C'est superflu AC/On. Ca fait 27, 18. Alors et nous ça fait combien ?
448. E1 : Nous aussi, on a pas AC/On n'est ce pas ?
449. E2 : Non, ce n'est pas nécessaire. Tu vois, au tableau il n'y pas AC/On, non plus.
450. E1 : Mais sans AC/ON comment on peut continuer à répéter ?
451. E2 : Non, je l'ai déjà essayé. Ca ne vaut pas la peine.
452. E1 : Mais on a besoin de la touche « = ».
453. Ppu : Vous regardez au tableau, s'il vous plaît. Vous avez des messages de groupes (suivant l'ordre dans lequel les messages sont collés au tableau) :

M1 :
Message : - 3 Sto A
Répéter ~~continuellement~~ groupe touches 164 fois
A + 0.03 Sto A
A × A + 1 =
Longueur du message : 23 touches
Liste de mots : Répéter ; ~~Continuellement~~ ; Groupe touches ; Fois

M2 :
3 Sto A
1 Sto B
$A \times A + 1 =$
$A + B \times 0.03 \ Sto \ A$ } Répéter encore 164 fois avec
$B + 1 \rightarrow B$
A + 0.03 Sto A 1 Sto B
Longueur du message : 37 touches
Liste de mots : Répéter encore ~~164~~ fois avec

M3 :
3 Sto A
1 Sto B
$A \times A + 1 =$
$A + B \times 0.03 \ Sto \ A$ } Répéter ~~pareil~~ 164 fois avec
$B + 1 \rightarrow B$
~~A = x + 0.03 x ∈ [-3 ; 2]~~ A + 0.03 Sto A 1 Sto B
Longueur du message : 37 touches
Liste de mots : Répéter ; fois ; avec

M4 :
3 Sto A A + 0.03 Sto A
A × A + 1 Sto B B × B + 1 Sto C C × C + 1
~~Répéter identiquement précédent avec B, C~~ Répéter encore depuis début
Longueur du message : 166
Liste de mots : ~~Répéter identiquement précédent avec B, C~~ Répéter encore depuis début

M5 :
3 Sto A
A + 0.03 Sto A
A × A + 1 =
~~Répéter 166 fois 2 calculs suivants~~
Répéter 166 fois 2 derniers calculs
Longueur du message : 27
Liste de mots : Répéter, fois, calcul, dernier

M6 :
3 Sto A
1 Sto B
A × A + 1
A + B × 0.03 Sto A
B + Sto B
Répéter ~~encore~~ 166 fois 3 derniers calculs
Longueur du message : 32
Liste de mots : Répéter, fois, calcul, dernier

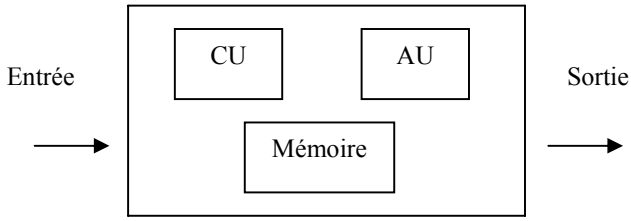
M7 :
- 3 Sto A A × A + 1 =
Taper ~~répéter cent six cinq~~ 165 fois ~~message suivant~~ : message suivant
A + 0.03 Sto A
A × A + 1 =
Longueur du message : ~~29~~ 32
Liste de mots : ~~Répéter cent six cinq~~, taper 165 fois message suivant

M8 :
3 Sto A ~~AC/ON~~
Répéter encore depuis A + 0.03 Sto A ~~AC/ON~~
A × A + 1 =
Longueur du message :
Liste de mots : Répéter encore depuis

454. E (en observant les messages) : Pourquoi ils n'ont besoin que A × A + 1 ? Alors notre groupe, on a fait des choses superflues.

455. E (du même binôme) : Pourquoi on doit mémoriser la touche B ? En tout cas l'expression est $x^2 + 1$.
456. Ppu (après avoir collé tous les messages) : S'il vous plaît. Pour le groupe 1, on doit répéter 164 fois. Quelle est la longueur du message ?
457. Es : 23.
458. Ppu : Vous voyez -, 3, Sto, (il compte publiquement les mots dans le message) Ca fait 25.
459. E : Oh, il manque que A × A + 1.
460. Ppu : Vous voyez c'est ça A × A + 1.
461. E : Mais pourquoi c'est différent de nous ? Nous on en a trouvé 27.
462. Ppu : la liste de mots, vous voyez c'est « Répéter ; groupe touches ; fois » donc ça fait 5. Alors, passons au 2e message, ici ce groupe s'appuie sur le 2e groupe de touches et ils ont trouvé 37 touches. Ce message est donc plus long. Et les mots qu'ils ont proposés est « répéter ; encore ; fois et avec ». Donc 4 mots. Le 3e message est pareil que le 2e. Il y a aussi 37 touches.
463. Es : Ils se sont recopiés alors.
464. Ppu : Le 4e message, les mots qu'ils ont proposés sont « Répéter ; encore ; depuis début ». Mais pourquoi la longueur du message est 166 ? Pourquoi ? les autres messages sont assez semblables que ceux qu'on vient de voir. Je veux vous faire deux remarques. D'abord en ce qui concerne la longueur du message. Tout à l'heure vous l'avez trouvé combien ? Quelques milles n'est ce pas ?
465. Es : 2 milles 3 cents. 2 milles.
466. Ppu : Oui, vous voyez environ 2 milles. N'est-ce pas ? Alors que maintenant on l'a réduit à environ 30 touches. Pour quoi on peut le réduire autant ?
467. Es : Le robot Calculator peut maintenant comprendre le vietnamien.
468. Ppu : Car vous lui attribuez une nouvelle capacité qui est de savoir répéter et la capacité de contrôler la répétition. Donc s'il comprend, il sait répéter et il sait contrôler la répétition alors vous pouvez réduire considérablement le message. Donc pour cette séance, ce que je voudrais que vous reteniez, ce sont les premières notions d'Informatique. Lorsque vous apprenez la programmation, ce sont aussi les premières notions. Donc pour ceux qui poursuivront en Informatique, en langage Pascal, ou pseudo Pascal qu'est ce qu'on peut écrire ? Alors on donne, pour ce message (affiche M1 et écrit en même temps au tableau) – 3 on donne à quoi ? C'est à A et 1…Et pour répéter en Pascal on utilise le mot « repeat » en anglais. Et A fois A plus 1 puis A + 0.03 Sto A. Et répéter…en Pascal ça veut dire qu'on répète ça 164 fois.
Au tableau :
3 Sto A
Repeat 164
A + 0.03 Sto A
A × A + 1 =
469. Ppu : Et lorsque vous écrivez les premiers programmes c'est comme ça. Vous vous rendez compte aussi qu'à partir d'un très long programme, on l'a réduit à un si court programme grâce aux amélioration de la capacité du robot, c'est-à-dire l'ordinateur. Donc par là je veux vous dire la 2e chose. Au début lorsque vous calculez l'expression, la calculatrice Alpro contient uniquement une chose pour calculer (il commence à dessiner au tableau). Cette partie est appelée, en langage de l'architecture d'ordinateur, unité arithmétique ; je le note AU, c'est-à-dire « Arithmetic Unit ». Donc dans la question 1, vous vous servez seulement de ça d'Alpro. Et puis dans la suite, vous avez la mémoire, c'est-à-dire les touches mémoires A, B, C, n'est-ce pas ? Et dernièrement, vous avez ajouté au robot Calculator la capacité de répéter. C'est-à-dire vous lui avez ajouté unité de contrôle pour contrôler la répétition. Vous comprenez ? Donc vous avez CU, ça veut dire « Contrôle Unit ». Donc ces 3 composants constituent l'essentiel de l'architecture de l'ordinateur : unité arithmétique, je note au passage qu'auparavant on fabriquait des machines qu'on faisait tourner à la main, comme les machines arithmétiques de Pascal, on n'avait que cette partie-là. Et puis la mémoire qui y est ajoutée et avec ça, en même temps on a l'unité de contrôle. Quelque soit la complexité d'une machine, ces trois parties sont fondamentales. Bien sûr il faut ajouter une partie pour faire entrer les données donc c'est l'entrée et ici on peut sortir sur l'écran ou sur l'imprimante donc la sortie.
470. Ppu : Donc deux choses que je voudrais que vous reteniez après cette séance, la première concerne la programmation et la deuxième est à propos de l'architecture de l'ordinateur. J'espère que cela vous sera utile et vous donne certaines idées. Malgré que quelques élèves ne puissent pas venir, je vous remercie de votre coopération.

Figure au tableau :



II. Des copies de binômes

II.1. Copie du binôme de la classe 1er F : des mémoires variables aux mémoires effaçables

Pierre et Bella. **SÉANCE 3**

Phase 3 (15 minutes)

Travail en binôme avec 1 transparent, feutre et Alpro (15 minutes)

Transparent du binôme :

Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.

Comment le lui dire ?

On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.

Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II.

CALCULATOR II, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$

Message : - 3 STO A
Répéter 166 fois A x A + 1 =
A + 0,03 STO A

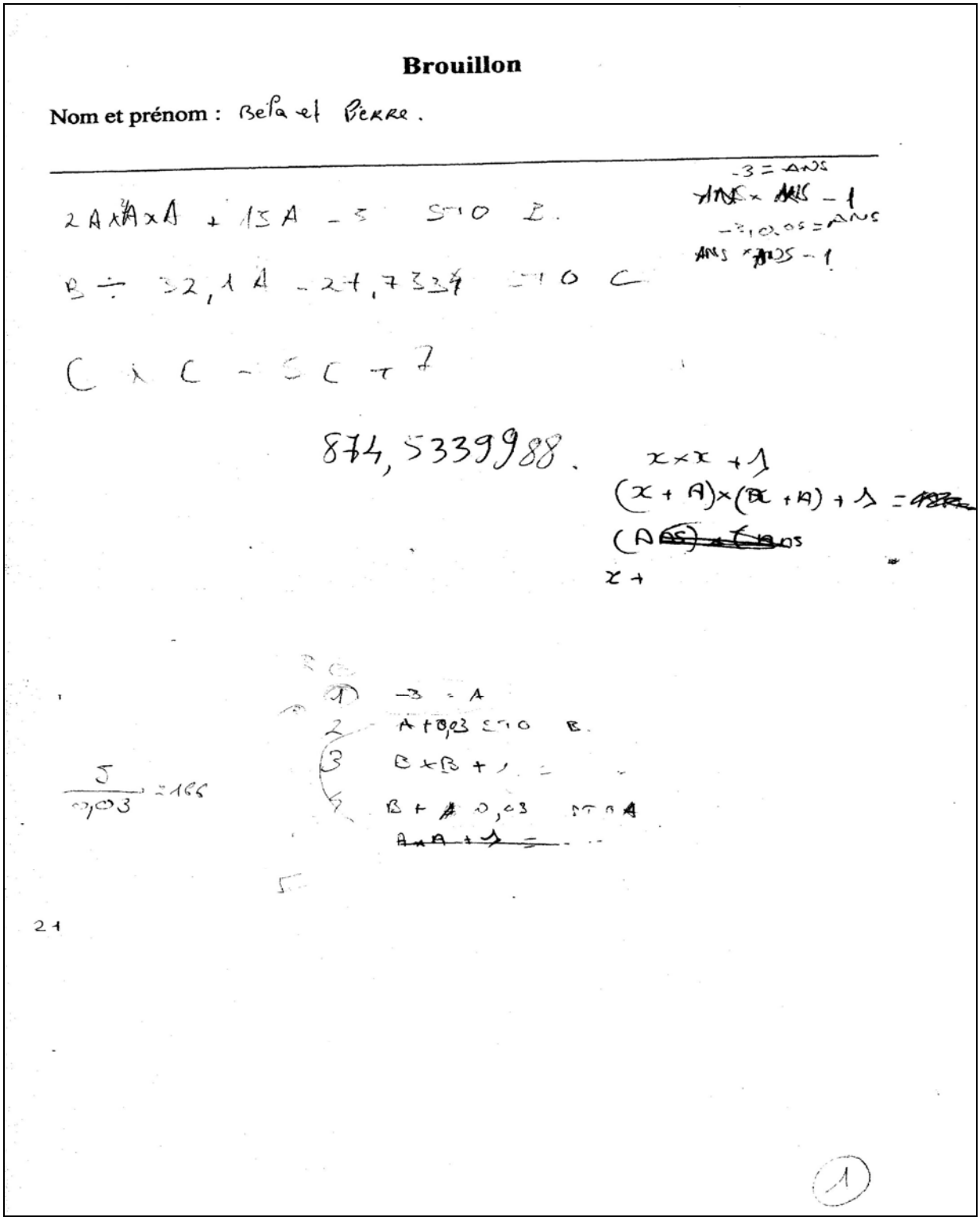
-3 STO A
Répéter 166 fois A x A + 1 =
A + 0,03 STO A

Longueur du message : 23

Pour calculer la longueur du message, les mots seront comptés comme les touches.

Liste de mots :
répéter
fois

II.2. Brouillons du binôme de la classe 1er F : des mémoires variables aux mémoires effaçables



Brouillon

Nom et prénom : Bella et Pierre.

Brouillon

Nom et prénom : Béla et Pierre. 1

~~-3 STO A A = -3,003~~ -3,003 STO A

A + 0,03 -> STO B A + 0,03 -> STO B

~~B x B + 1 = ...~~

~~B + 0,03 -> STO A~~

~~A x A + 1 = ...~~

~~A + 0,03 -> STO B~~

-3 + 0,03

ANS x ANS + 1 =

-3

-3 x -3 + 1 =

-3

-3+0,03

~~-2,97 + 0,~~

A = -3,003

-3

4

Répéter 166 fois A x A + 1 =
A + 0,03 STO

En partant de -3 STO A

-3 STO A

Répéter 166 fois A x A + 1 =
A + 0,03 STO A

II.3. Copie du binôme de la classe 1er V : de la mémoire Ans aux mémoires effaçables

Le vieux cinq
Nai Tất Đạt.

Tiết 4

Pha 3 (15 phút)
Làm việc theo nhóm, dùng giấy trong, bút phớt và máy tính Alpro

Giấy trong của nhóm :

Ngoài các khả năng của CALCULATOR, người máy được nâng cấp CALCULATOR II phải lặp lại được sự thực hiện của một nhóm phím của Alpro được viết trên bảng.
Làm thế nào để nói điều đó với người máy ?
Ta giả sử rằng CALCULATOR II có thể hiểu nhiều nhất là năm từ của tiếng Việt giúp điều hành việc lặp sự thực hiện một nhóm phím của Alpro.

Các em phải chọn các từ này và viết một thông báo ngắn nhất cho CALCULATOR II.

Khi thực hiện thông báo này CALCULATOR phải in ra *tất cả các giá trị của hàm số tương ứng với các giá trị của x thuộc đoạn [- 3 ; 2] và chỉ các giá trị này mà thôi,* khoảng cách giữa hai giá trị liên tiếp của x bằng 0,03.
Nhắc lại $f : x \to f(x) = x^2 + 1$

Thông báo :

- 3 sto A
Lặp ~~liên tục~~ nhóm phím 164 lần
A + 0,03 sto A
A x A + 1 =

Độ dài thông báo : 23 phím

Để tính độ dài của thông báo, các từ được tính như các phím.

Danh sách các từ :

Lặp
~~liên tục~~
nhóm phím
lần

II.4. Copies de quelques binôme des classes 1er F et 1er V

SÉANCE 3

Phase 3 (15 minutes)

Travail en binôme avec 1 transparent, feutre et Alpro (15 minutes)

Transparent du binôme : Hall B.

Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.

Comment le lui dire ?

On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.

Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II.

CALCULATOR II, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$

Message :

- 3,03 + 0,03 sto A
A x A + 1 =
A + 0,03 sto A
refaire avec la nouvelle valeur A.

Longueur du message : 32

Pour calculer la longueur du message, les mots seront comptés comme les touches.

Liste de mots :

- refaire
- avec
- la
- nouvelle
- valeur

SÉANCE 3

Phase 3 (15 minutes)

Travail en binôme avec 1 transparent, feutre et Alpro (15 minutes)

Transparent du binôme : 2

Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.

Comment le lui dire ?

On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.

Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II.

CALCULATOR II, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$

Message :

-3 STO A ⓪
→ A x A + 1 = ①
└ A + 0,03 STO A ②

Etape ⓪ ① ② ① etc

Longueur du message :

Pour calculer la longueur du message, les mots seront comptés comme les touches.

Liste de mots :

Lang Huang.

SÉANCE 3

Phase 3 (15 minutes)

Travail en binôme avec 1 transparent, feutre et Alpro (15 minutes)

Transparent du binôme :

Le robot amélioré, CALCULATOR II, en plus des capacités de CALCULATOR, doit pouvoir répéter l'exécution du groupe de touches d'Alpro écrit au tableau.

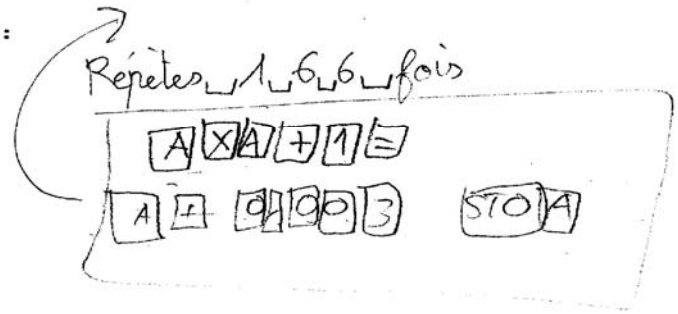
Comment le lui dire ?

On imagine que CALCULATOR II peut comprendre au maximum cinq mots de la langue française qui aident à contrôler la répétition de l'exécution d'un groupe de touches d'Alpro.

Vous devez choisir ces mots et écrire un message le plus court possible à CALCULATOR II.

CALCULATOR II, en exécutant ce message, doit *imprimer toutes les images par la fonction f des valeurs de x appartenant à l'intervalle [-3 ; 2] et seulement ces images*, l'écart entre deux valeurs successives de x étant égal à 0,03. Rappel : $f : x \rightarrow f(x) = x^2 + 1$

Message :



Longueur du message : 20 touches

Pour calculer la longueur du message, les mots seront comptés comme les touches.

Liste de mots :

Répéter

fois

20/20

Tiết 4

Pha 3 (15 phút)

Làm việc theo nhóm, dùng giấy trong, bút phớt và máy tính Alpro

Giấy trong của nhóm : Kiên + Linh

Ngoài các khả năng của CALCULATOR, người máy được nâng cấp CALCULATOR II phải lặp lại được sự thực hiện của một nhóm phím của Alpro được viết trên bảng.

Làm thế nào để nói điều đó với người máy ?

Ta giả sử rằng CALCULATOR II có thể hiểu nhiều nhất là năm từ của tiếng Việt giúp điều hành việc lặp sự thực hiện một nhóm phím của Alpro.

Các em phải chọn các từ này và viết một thông báo ngắn nhất cho CALCULATOR II.

Khi thực hiện thông báo này CALCULATOR phải in ra *tất cả các giá trị của hàm số tương ứng với các giá trị của x thuộc đoạn [- 3 ; 2] và chỉ các giá trị này mà thôi,* khoảng cách giữa hai giá trị liên tiếp của x bằng 0,03.

Nhắc lại $f : x \to f(x) = x^2 + 1$

Thông báo :

-3 STO A
| A + 0,03 STO A
| A x A + 1 =
Lặp 166 lần 2 phép tính ~~sao~~
Lặp 166 lần 2 phép tính cuối

Độ dài thông báo : 27

Để tính độ dài của thông báo, các từ được tính như các phím.

Danh sách các từ :

Lặp
lần
phép tính cuối

Tiết 4

Pha 3 (15 phút)

Làm việc theo nhóm, dùng giấy trong, bút phớt và máy tính Alpro

Giấy trong của nhóm : Hương, Trung

Ngoài các khả năng của CALCULATOR, người máy được nâng cấp CALCULATOR II phải lặp lại được sự thực hiện của một nhóm phím của Alpro được viết trên bảng.

Làm thế nào để nói điều đó với người máy ?

Ta giả sử rằng CALCULATOR II có thể hiểu nhiều nhất là năm từ của tiếng Việt giúp điều hành việc lặp sự thực hiện một nhóm phím của Alpro.

Các em phải chọn các từ này và viết một thông báo ngắn nhất cho CALCULATOR II.

Khi thực hiện thông báo này CALCULATOR phải in ra *tất cả các giá trị của hàm số tương ứng với các giá trị của x thuộc đoạn [- 3 ; 2] và chỉ các giá trị này mà thôi,* khoảng cách giữa hai giá trị liên tiếp của x bằng 0,03.

Nhắc lại $f : x \rightarrow f(x) = x^2 + 1$

Thông báo :

-3 STOA
1 STOB
A x A + 1
A + B . 0,03 STOA
B + 1 STOB
Lặp lại 166 lần và 3 phép tính cuối

Độ dài thông báo : 32

Để tính độ dài của thông báo, các từ được tính như các phím.

Danh sách các từ :

Lặp
lần
phép tính cuối

6 1/2

Tóm tắt

Giữa Toán học và Tin học có một sự gắn bó mật thiết trong lịch sử phát triển chúng cũng như trong việc nghiên cứu chúng hiện nay. Bằng chứng là sự trợ giúp thường xuyên của các thuật toán trong việc giải các bài toán cơ bản cũng như sự tồn tại của môn lý thuyết thuật toán như là một lĩnh vực đặc thù của Tin học bên cạnh các lĩnh vực khác như lí thuyết ngôn ngữ hay như lí thuyết ô-tô-mát.

Nghiên cứu của chúng tôi quan tâm đến vấn đề dẫn nhập các yếu tố của lý thuyết thuật toán và lập trình trong dạy học toán phổ thông. Nghiên cứu này dựa trên các phân tích tri thức luận và phân tích thể chế. Một mặt các phân tích này chỉ ra rằng các khái niệm Tin học cơ bản như biến, vòng lặp được hình thành cùng một lúc với sự biến đổi của kiến trúc máy tính. Mặt khác chúng chỉ ra sự tồn tại một cách khó khăn của các yếu tố trên trong dạy học Toán phổ thông ở Pháp và Việt nam.

Các kết quả phân tích là cơ sở cho việc xây dựng, thiết kế và thực hiện một công nghệ didactic trong môi trường Tin học. Công nghệ này nhằm tới sự phát sinh có tính thực nghiệm về các khái niệm như máy tính Von Neumann, lập trình qua việc viết các thông báo liên tiếp (các chương trình) cho các máy tính có các đặc tính khác nhau. Việc thiết kế này sử dụng các công cụ của lí thuyết Tình huống Didactic và nhờ vào một tình huống cơ sở của lý thuyết thuật toán và lập trình để tổ chức các cuộc gặp gỡ với các loại bộ nhớ khác nhau của máy tính bỏ túi. Chúng tôi đặc biệt chú trọng loại bộ nhớ xoá được vì chúng cho phép sự hình thành của các khái niệm biến Tin học và vòng lặp trong công nghệ didactic này.

Do các nhu cầu nghiên cứu, chúng tôi đã thiết kế một máy tính mô phỏng, tên là Alpro, dựa trên các kiểu máy tính tồn tại trong dạy học phổ thông ở hai nước và có thêm khả năng phụ là ghi nhớ tất cả các phím được bấm trong quá trình tính toán.

Từ khóa

Phân tích tri thức luận và thể chế, lí thuết Tình huống Didactic và công nghệ didactic, phát sinh thực nghiệm, bộ nhớ xoá được, chương trình ghi, kiến trúc máy tính, bài toán lập bảng hàm số thực, thuật toán, lập trình một giải thuật, biến Tin học, vòng lặp

Résumé de la thèse

Il y a entre les mathématiques et l'informatique une solidarité fondamentale qui repose sur l'histoire et sur les pratiques actuelles de ces disciplines. Une preuve en est le recours constant aux algorithmes dans les résolutions de problèmes mathématiques fondamentaux et l'existence de l'algorithmique comme domaine constitutif de l'informatique aux côtés d'autres comme la théorie des langages ou la théorie des automates.
Notre recherche étudie la question de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire. Elle s'appuie sur des analyses épistémologique et institutionnelle qui montrent d'une part que les notions de boucle et de variable se construisent en même temps que l'architecture de la machine se transforme et atteste d'autre part de la vie difficile d'éléments d'algorithmique et de programmation dans l'enseignement secondaire en France et au Viêt-nam.
Les résultats de ces analyses fondent la conception et la réalisation d'une ingénierie didactique dans un environnement informatique conçue comme une genèse expérimentale de la machine de Von Neumann et de la programmation à travers l'écriture des messages successifs (programmes) à des machines dotées de caractéristiques différentes. Cette conception utilise les outils de la théorie des Situations Didactiques pour organiser, à partir d'une situation fondamentale de l'algorithmique et de la programmation, une première rencontre avec différents types de mémoires d'une calculatrice en particulier les mémoires effaçables, permettant l'émergence de la notion de variable informatique et de boucle dans notre ingénierie.
Pour le besoin de notre recherche, nous avons construit un émulateur de calculatrice, nommé Alpro, sur le modèle des calculatrices existantes dans l'enseignement secondaire des deux pays et ayant la capacité supplémentaire d'enregistrer l'historique des suites des touches appuyées lors d'un calcul.

Mots clés

Analyses épistémologique et institutionnelle, Théorie des Situations et ingénierie didactique, genèse instrumentale, mémoire effaçable, programme enregistré, architecture d'une machine, problème de tabulation d'une fonction numérique, algorithme, programmation d'un algorithme, variable informatique, itération.