Introduire des éléments d'algorithmique et de programmation dans l'enseignement secondaire ? Une étude didactique

Nguyễn Chí Thành
Faculté de l'Education, Université Nationale du Vietnam à Hanoi

Annie Bessot
IREM de Grenoble et Laboratoire LIG, équipe DIAM, Université de Grenoble

Cet exposé a pour objectif de présenter le travail de thèse de Nguyen Chi Thanh (2005)[41].
Nous présentons d'abord les résultats d'une analyse épistémologique concernant l'algorithmique et la programmation. Puis nous donnons les résultats d'une analyse institutionnelle comparative France Viêt-nam en ce qui concerne la présence d'éléments informatiques dans l'enseignement mathématique secondaire (noté par la suite EMS) sous les programmes autour des années 2000. Nous interrogeons brièvement les nouveaux programmes 2009. Nous terminons en présentant les éléments d'une ingénierie didactique expérimentée en France et au Viêt-nam : conception, réalisation et analyse dans les conditions et les contraintes du programme en vigueur lors de l'expérimentation.

Analyse épistémologique

1. Analyse de la genèse de la machine ordinateur en rapport avec les problèmes mathématiques

Nous nous attachons à repérer les ruptures et les filiations dans la co-genèse de la notion de machine ordinateur et de la programmation d'algorithmes, en prenant en compte 3 critères :
- les problèmes mathématiques, ayant une solution algorithmique connue,
- les savoirs mathématiques de l'époque en relation avec ces problèmes
- le contexte technologique de l'époque
Nous portons aussi notre attention sur l'évolution d'une étape à une autre du rapport de l'homme avec la machine.

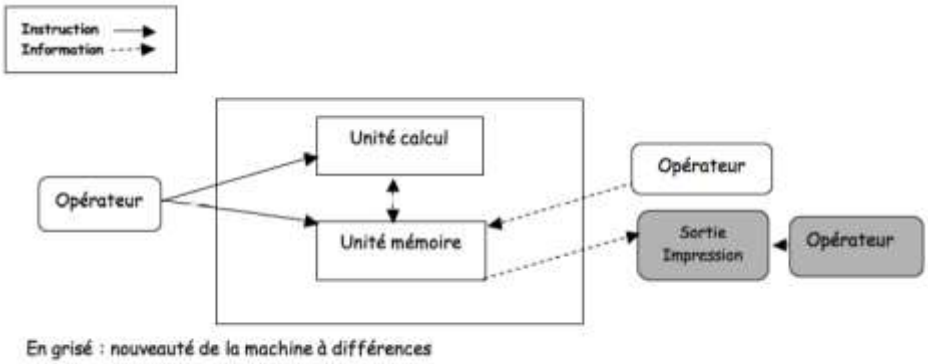
1.1. La machine arithmétique

La mécanisation des calculs fut d'abord celle des algorithmes liés aux 4 opérations de l'arithmétique décimale. *L'enjeu initial* de cette mécanisation du calcul est bien décrit par Leibniz : se libérer du calcul qu'il qualifie 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

[41] Thèse en cotutelle France – Viêt-nam codirugé par Anie Bessot et Lê Van Tien. Philippe Jorrand en a contrôlé les parties informatiques. Alain Birebent a aussi participé à ce travail.

être confié à n'importe qui, avec l'aide de machine. » (Leibniz cité par Ligonnière 1987, p. 41)
Cette mécanisation s'appuie sur l'existence d'un contexte technologique, celui du mécanisme d'horlogerie avec roues dentées et ergots
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 de Pascal (1642) et la machine de Leibniz (1672 - 1693) : les dispositifs, qui servent à inscrire les deux nombres pour l'opération et à afficher le résultat, sont dissociés.
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 la tabulation de toute fonction polynomiale et par approximation celles d'une classe plus large de fonctions.
Charles Babbage, mathématicien anglais (1791-1871), 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 : *second enjeu de la mécanisation des calculs*. La première machine de Babbage « difference engine » (1821) 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)
Mais la seule innovation par rapport aux machines précédentes du projet de machine à différences est d'imaginer l'impression automatique des résultats. 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 de la machine ordinateur actuelle.

Nous représentons par un schéma l'architecture de la machine arithmétique (cf. figure



1).

Figure 1. Architecture de la machine arithmétique

Dans la machine arithmétique, 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 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.
Chaque machine arithmétique est un programme mécanisé : le programme est interne et figé.

1.2. Une première rupture : la machine analytique de Babbage

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 (1834-1836).
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 de 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, suite aux 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'équations (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.
Grâce aux cartes perforées, 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.
En plus de la faculté de décision, Babbage conçoit pour la première fois la notion de mémoire (« store ») structurée en colonnes et ayant la propriété d'être effaçable.
Cette machine n'est pas construite quand Ada Lovelace écrit un premier programme informatique. Cette écriture nécessite un travail mathématique préalable. Ce travail mathématique lui permet à la fois d'identifier un invariant de cycle et la formulation d'une condition d'arrêt (compteur décrémenté).

Pour nous faire comprendre, examinons le problème du calcul des nombres Bernoulli .
Ci-après le travail mathématique d'Ada Lovelace avant l'écriture d'un programme à la machine Analytique de Babbage.

Ada Lovelace part d'une formule connue

$$\frac{x}{e^x-1} = 1 - \frac{x}{2} + B_1\frac{x^2}{2} + B_3\frac{x^4}{2.3.4} + B_5\frac{x^6}{2.3.4.5.6} + \dots$$

Elle transforme cette formule en la formule de récurrence suivante :

$$-\frac{1}{2}\cdot\frac{2n-1}{2n+1} + B_1\frac{2n}{2} + B_3\frac{2n.(2n-1).(2n-2)}{2.3.4} + \dots + B_{2n-1} = 0$$

avec n entier supérieur à 0 et où $A_0 + A_1B_1 + A_3B_3 + \dots B_{2n-1} = 0$

Ce travail mathématique lui permet de transformer une formule initiale contenant une infinité dénombrable de valeurs en un processus de calcul n-fini.
Ce processus est basé sur une double formule de récurrence : 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} ($j = 1 \dots i-1$)
Il permet aussi d'attacher la condition d'arrêt du processus de calcul à la valeur finie n.
Ada Lovelace écrit alors un programme pour n = 4 et l'exécute « à la main ».
Nous voyons donc que l'écriture d'un programme (qui devient externe) s'accompagne d'un retour réflexif sur les objets mathématiques de la formule afin de dégager la condition d'arrêt et l'invariant d'un cycle.

Une colonne variable devient alors un emplacement dans l'unité de mémoire effaçable dans *lequel on stocke intentionnellement une* donnée, c'est-à-dire une variable informatique.

Nous représentons par un schéma l'architecture de la machine Analytique de Babbage (cf. figure 2).

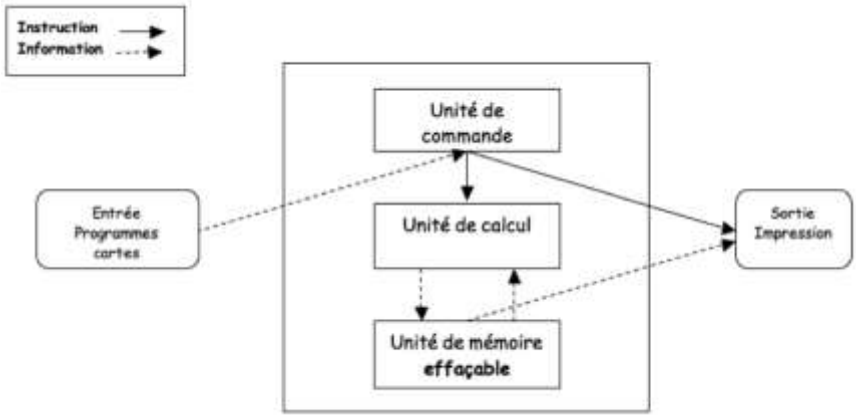


Figure 2. Architecture de la machine Analytique

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. De plus, la mémoire de cette machine a pour propriété fondamentale d'être *effaçable* contrairement à la mémoire des machines arithmétiques. Nous avons montré comment Ada dans le premier programme à cette machine utilise ce caractère d'effaçabilité de la mémoire pour faire émerger les notions de variable et de boucle.
Mais cette machine à programmes externes a des limites.
Les programmes sont contenus 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 les 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)

1.3. Une deuxième rupture : la machine ordinateur 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. 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 : en ce sens cette machine se différencie fondamentalement des machines calculateurs et prend le nom d'ordinateur.
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 .
Cela va entraîner un accroissement considérable du champ des problèmes calculables.
En travaillant sur le 10e 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. Ce travail va conduire à la notion de problème décidable : 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 effective à ce problème. Les années 30 ont consacré l'existence de problèmes indécidables ce qui a nécessité de définir la notion d'algorithme. Nous retiendrons la définition attachée à la machine hypothétique de Turing (qui ne s'arrête jamais !) : Turing définit la notion d'algorithme comme un ensemble d'instructions pour sa machine simple. (Goldschlager et al., 1986). La thèse de Church-Turing postule 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.
L'architecture de tous les ordinateurs existants actuellement 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.
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'avènement de langages évolués proches de l'algorithme que l'on programme 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 ».
Nous n'en dirons pas plus dans le cadre de cet exposé : on peut se reporter à la thèse de Nguyen Chi Thanh pour une analyse de l'évolution des langages du programme d'Ada aux langages évolués.

Nous représentons par un schéma l'architecture de la machine ordinateur (cf. figure 3).

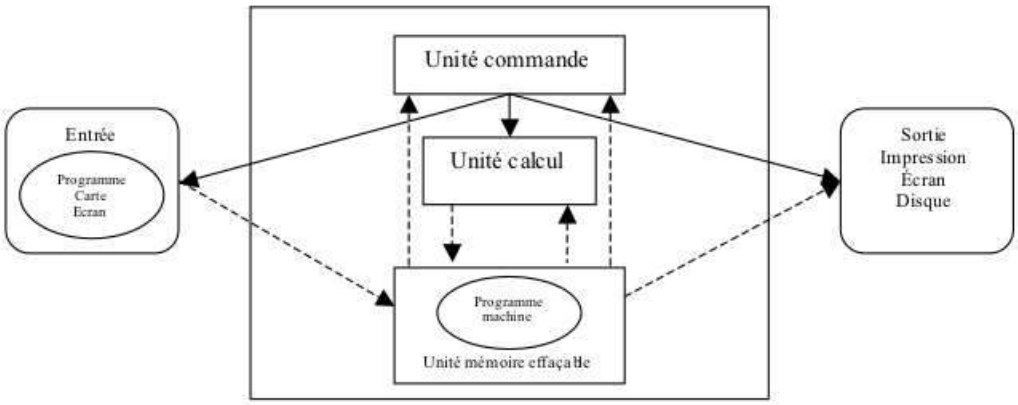


Figure 3. Architecture de la machine ordinateur

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 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 ces 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.
La définition de mémoire effaçable servira de référence pour la suite de l'exposé, en particulier pour la notion de variable informatique : 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.
L'idée qu'un programme dit enregistré et des données puissent partager une mémoire commune (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.

1.4. En conclusion, quelques définitions

Je vais m'appuyer sur un schéma pour conclure cette première partie de l'analyse épistémologique (cf. figure 4).

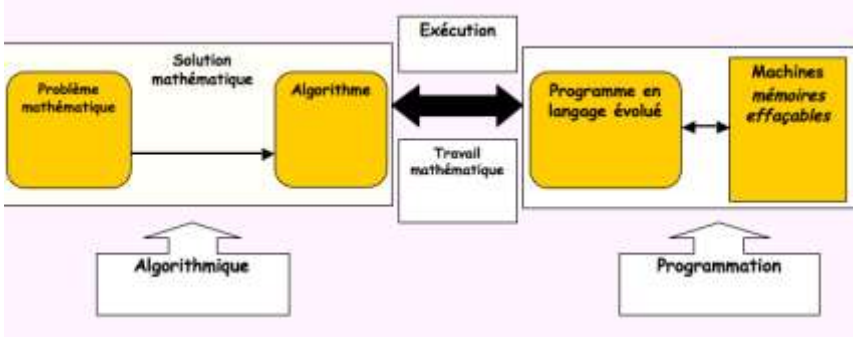


Figure 4. Schéma des rapports entre algorithmique et programmation en mathématiques

Un algorithme est une suite finie d'instructions à appliquer dans un ordre déterminé à un nombre fini de données, pour arriver avec certitude, en un nombre fini d'étapes, à un certain résultat, *solution d'un problème mathématique*, et cela indépendamment des données. (Bouvier et al. 1993)
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)
Un programme informatique est une liste des instructions auxquelles *la machine* devra obéir, dans l'ordre de leur exécution […]. On le chargera dans la mémoire de la machine, où elle puisera les instructions au fur et à mesure de leur exécution, à sa propre vitesse (Arsac 1995)
Les langages de programmation permettent de définir les ensembles d'instructions effectuées par l'ordinateur lors de l'exécution d'un programme. Un programme est donc l'expression d'un *algorithme* dans un langage donné pour une *machine* donnée.
L'articulation entre algorithmique et programmation se construit en même temps que l'architecture des machines évolue

2. Émergence d'un enseignement universitaire d'algorithmique et de programmation

Nous présentons très brièvement maintenant comment a émergé un enseignement d'algorithmique et de programmation en même temps qu'a émergé l'informatique comme discipline scientifique à l'université. Nous chercherons à dégager les conditions mises en place par ces pionniers et des stratégies possibles d'enseignement. Nous accorderons une importance particulière à la place de la machine dans ces enseignements.
Pour repérer l'arrivée dans l'enseignement universitaire d'objets fondamentaux de l'informatique (centration sur les notions de boucle et variable), nous avons choisi des moments clés marqués par trois traités. *Un traité est un ouvrage écrit par un auteur, qui non seulement contribue à la production d'un domaine ici l'informatique mais aussi à son enseignement.*
Ces traités sont les suivants :
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 pour l'algorithmique « Fundamentals of Computer Algorithms » de Horowitz et Sahni.
Remarquons tout d'abord que l'enseignement de la programmation a précédé l'enseignement de l'algorithmique.
Quelles conditions sont mises en place dans ces traités pour cet enseignement et quelle place y occupe la machine et les langages de programmation ?
- *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 mais évoquée puisque absente de l'environnement des étudiants. Les objets informatiques fondamentaux présents sont instructions, boucles, programmes,

organigrammes et architecture de la machine. 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.
- *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.
Knuth développe un discours explicatif sur la notion de variable, au travers de la notion d'affectation ; en particulier il prend soin de distinguer son usage de celle de variable mathématique. La description d'un algorithme par un organigramme avec son système de règles d'écriture établit la notion de boucle
- *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. 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.

L'analyse de ces trois traités permet de dégager deux grandes stratégies d'enseignement.
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 construites mais particulière comme celle de Babagge, 1842), ou idéales (c'est-à-dire non construites mais représentantes 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.*
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.
Les traités de Kuntzmann et de Knuth relèvent de la première stratégie, celui d'Horowitz et Sahni de la deuxième. La deuxième stratégie est la stratégie dominante de l'enseignement actuel d'algorithmique.

Nous allons maintenant présenter les points principaux d'une analyse comparative de deux systèmes d'enseignement : EMS en France autour des années 2000 et au Viêt-Nam dans les années 1990 - 2000.

Analyse institutionnelle comparative des EMS France, Viêt-nam[42]

1. Avant 2009

ll y a volonté institutionnelle d'introduire des éléments d'informatique dans les deux institutions, Lycée en France et au Viêt-nam, mais avec deux points de vue différents.
En France, cette introduction se fait au sein des mathématiques dans des domaines, comme l'analyse, la statistique et à partir de 2000 l'arithmétique, sans que la notion d'algorithme soit objet d'enseignement.
Au Viêt-nam, l'introduction est tentée à deux endroits : en mathématique et en informatique laquelle est considérée comme une discipline autonome.
Par un examen de manuels en France pour le programme 2000 et ceux en vigueur au Viêt-nam entre 1990 et 1998, Nguyen Chi Thanh a tenté de répondre aux questions suivantes :
- Y a-t-il présence d'algorithmes ? Si oui, quel rôle jouent-ils ?
- Y a-t-il présence de programmes informatiques ?
- Comment vivent-ils ?

Nous considérons les manuels Belin, Déclic, Bréal en France - M1, M2, M3 au Viêt-nam

Regardons le nombre d'exercices qui concernent la notion de variable informatique. Comme le montre le tableau ci-après les exercices sont quasi inexistants dans les deux EMS (cf. tableau 1).

Belin	3	M1	0
Déclic	2	M2	4
Bréal	0	M3	0

Tableau 1. Effectif des exercices sur la notion de variable informatique

Comment l'élève peut-il différentier la notion de variable informatique (qui désigne une case de mémoire) de la notion préconstruite de variable mathématique ?

De la même façon examinons les exercices concernant l'écriture d'un algorithme : ils sont inexistants en France, nombreux au Viêt-nam mais là les algorithmes ne sont pas itératifs (cf. tableau 2)

Belin	1	M1	2
Déclic	2	M2	19
Bréal	0	M3	5

Tableau 2. Effectif des exercices sur l'écriture d'algorithme

[42] Au Viêt-nam, Nguyen Chi Thanh s'est appuyé sur la recherche de Le Van Tien (2001), qu'il a complété par l'analyse de trois collections de manuels (1990) (M1), (M2) et (M3).

Les exercices d'exécution d'un programme sont nombreux en France, mais avec une très faible familiarisation des élèves avec les algorithmes mathématiques ; ils sont par contre quasi inexistant au Viêt-nam (cf. tableau 3).

Belin	33	M1	1
Déclic	22	M2	3
Bréal	11	M3	2

Tableau 3. Effectif des exercices sur l'écriture d'algorithme

En conclusion, des algorithmes mais pas d'algorithmique enseignée ; des programmes mais pas de programmation enseignée.
Soulignons de plus l'absence institutionnelle de tout langage de programmation aussi bien en France qu'au Viêt-nam.

Dans les deux institutions, on peut faire un constat de grandes difficultés voire d'échec de ces tentatives à des degrés divers.
Au Viêt-nam, le chapitre informatique a été supprimé et une deuxième tentative (introduction de l'informatique en tant que discipline) a, elle aussi, été éphémère (1995-96).
En France, sous les programmes 2000, si les notions de base en informatique se maintiennent, elles vivent mal : cela se traduit dans les manuels par l'effacement de la nature algorithmique des procédés d'approximation, par le refus de construire des tâches relatives à la programmation des algorithmes et enfin par le recours à des logiciels de calcul pour instrumenter les techniques associées à ces tâches.
On peut avancer certaines raisons explicatives de cet échec ou difficultés.
Au Viêt-nam, le chapitre informatique est un « isolat » dans EMS 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 lieux 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 promoteurs, 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 de l'ordinateur) qui sont préconisés, prennent en charge les calculs itératifs et une part de l'élaboration de ces algorithmes.

2. Et dans les programmes 2009 en France ?

Pour questionner brièvement les programmes 2009, nous allons d'abord donner certaines idées essentielles issues du rapport de la commission Kahane de réflexion sur l'enseignement des mathématiques (2001), l'une des références du travail de Nguyen Chi Thanh.
Cette commission défend l'idée d'introduire une part d'informatique dans l'enseignement des mathématiques et dans la formation des maîtres en faisant évoluer progressivement les contenus pour intégrer de nouveaux objets et notions d'algorithmique et de programmation.

Ce rapport souligne une distinction fondamentale :
- d'une part, l'utilisation des ordinateurs et calculatrices et des logiciels qui y sont implantés,
- d'autre part , l'apprentissage et l'enseignement des concepts de base de l'algorithmique et de la programmation.
Les concepts de base de la programmation et de l'algorithmique sont pour la commission:
- structures de contrôle (*boucles et branchements*) et *récursivité* pour la programmation :
- *structures de données* et *complexité* pour la notion d'algorithme.
Un survol rapide des programmes 2009 pour la classe de seconde montre une certaine influence du travail de cette commission.
Par exemple, ce programme différencie fondamentalement l'algorithmique de l'usage des logiciels. Mais l'algorithmique est première et séparée de la programmation.
Il y a bien présence de certains des concepts de base de l'algorithmique et de la programmation identifiés par la commission Kahane mais ces concepts deviennent des compétences.
La complexité des algorithmes commande la machine. Les machines sont bien mentionnées explicitement, mais comme un environnement de programmation.
Il resterait à analyser les manuels actuels pour avoir une idée des pratiques possibles pour les enseignants et les élèves dans EMS sous les contraintes et conditions de ces nouveaux programmes.

Nous allons maintenant essayer de vous faire rentrer dans la conception et la réalisation de l'ingénierie didactique de Nguyen Chi Thanh.

Principaux éléments de la conception de l'ingénierie didactique

1. Une situation fondamentale possible de l'algorithmique et de la programmation

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 pour la résolution de problèmes mathématiques. Parmi ces problèmes nous choisissons le problème de tabulation d'une fonction numérique. Pourquoi ?
La résolution du problème de tabulation exige la répétition de calculs identiques. 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 ces machines. De plus ce problème est présent dans EMS, la notion de fonction étant une notion mathématique centrale au lycée.
C'est pour cela que nous formulons une situation fondamentale possible de l'algorithmique et la programmation pour l'ingénierie didactique qui articule un problème de tabulation à un problème informatique.
- Le *problème mathématique de la tabulation d'une fonction numérique* est formulé comme suit :

> « Soit **f** une fonction. Calculer les images par cette fonction de **m** nombres x_0, x_1…x_k…espacés d'un pas **p** et appartenant à l'intervalle **[a, b]** avec x_0 = a. »[43]

[43] en gras les choix possibles.

- Le *problème informatique est celui de l'écriture d'un message à une machine à mémoire effaçable* pour une exécution effective d'une solution du problème mathématique.
L'absence dans les programmes 2000 du Lycée de tout langage de programmation conduit au choix de la première stratégie d'enseignement de l'algorithmique repérée dans la partie épistémologique, à savoir la présence d'une machine de référence, réelle ou fictive avec comme enjeu la compréhension de l'architecture de la machine ordinateur et la conception d'un langage approprié à cette machine.
La machine de base des machines de l'ingénierie est la calculatrice non programmable. Ce choix est fondé par le fait que cette machine est présente dans la plupart des EMS en particulier celle du Viêt-nam.

2. Types de mémoires d'une calculatrice non programmable ?

Nous avons montré que le caractère d'effaçabilité d'une mémoire et le stockage intentionnel de données dans cette mémoire est nécessaire à l'émergence de la notion de variable informatique et de boucle. Rappelons que pour une telle mémoire le stockage d'une nouvelle valeur efface toute valeur déjà là. *Par abus de langage et par commodité, nous appellerons touches mémoires les touches de la calculatrice qui renvoient à une place dans l'unité de mémoire de la machine calculatrice.*
Nous examinons maintenant les principaux types de mémoires des calculatrices non programmables.

- Mémoires visibles

Quelles touches mémoires visibles de la calculatrice peuvent devenir candidates à donner du sens à la notion de variable informatique ?
• Un premier type de mémoire visible : les touches A, B, C, etc.
Deux opérations sont associés à ces touches : le stockage (Sto chez TI ou -> chez Casio) et le rappel (Enter chez TI ou = chez Casio ou Rcl dans les 2). Par exemple, si on fait la suite d'appuis suivante : « √3 sto A ; sin 30 =; A x 4 – 5 » alors la valeur dans la mémoire A est √3.
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 ».
• Un deuxième type de mémoire visible : la touche Ans
Cette deuxième mémoire se distingue de la « mémoire variable » par le fait que le stockage d'une donnée 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. Par exemple, si on fait la suite d'appuis « √3 = ; sin 30 = ; Ans x 4 – 5 », la valeur dans la mémoire est celle du dernier résultat à savoir sin 30.
La particularité d'effacer automatiquement le contenu rend la mémoire Ans impropre pour supporter la notion de variable informatique. La mémoire Ans est en quelque sorte la mémoire du dernier résultat. Nous dirons qu'elle est non effaçable.

- 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.

3. Une calculatrice générique : Alpro

Pour les besoins de l'ingénierie, nous avons conçu une calculatrice générique au sens où elle représente les calculatrices non programmables présentes en 2000 dans les EMS en France et au Viêt-nam. Ce projet avait en particulier pour objectif de répondre aux questions suivantes :

- Quelle est la pratique privée de la calculatrice chez les élèves ?
- Dans cette pratique privée quel est l'usage des mémoires ?

De plus nous voulions observer et contrôler la genèse instrumentale de ces mémoires. Nous avons donc construit un émulateur de calculatrice non programmable dit Alpro.
Une caractéristique de cet émulateur est l'enregistrement automatique d'un fichier de l'historique des touches actionnées : ainsi ces fichiers dits historiques nous donnent accès à ce que nous avons appelé « programme en actes » pour toute tentative de calcul effectif avec Alpro.
Deux versions de cet émulateur ont été construites : Alpro et Alpro bloqué que nous décrivons plus loin.

4. Des choix didactiques

Nous avons choisi d'écrire les nombres en écriture décimale parce que cette écriture 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. La notation Dj désignera désormais l'ensemble des nombres décimaux ayant j chiffres après la virgule.

- La machine de base de l'ingénierie est la calculatrice *Alpro bloqué* (cf. figure 5).

Figure 5. Image d'Alpro bloqué

Décrivons les restrictions apportées à Alpro et leurs raisons.

• On retrouve les deux types de mémoires potentiellement visibles d'une calculatrice non programmable : la Mémoire Ans et *trois* mémoires variables A, B, C. Nous avons choisi de restreindre les mémoires variables à trois (A, B et C) pour favoriser d'une part l'usage des mémoires variables et d'autre part 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.

• L'*indisponibilité* des touches *parenthèses* vise à 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.
• Nous avons choisi de ne conserver que les quatre opérations arithmétiques : « + », « - », « × », « ÷ ». Ce choix augmente, pour les multiplications en particulier (absence de la touche puissance), le coût des calculs et accroît les risques d'erreurs et justifie l'usage des mémoires.
• De plus l'émulateur donne accès à un message d'information concernant les touches mémoires quand le curseur se place 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.

Principaux éléments de la réalisation de l'ingénierie didactique

1. Les trois situations de l'ingénierie et leurs enjeux

L'ingénierie didactique est composée de 3 situations générées par la situation fondamentale et elle articule un problème mathématique de tabulation à un problème informatique qui est l'écriture de messages à différentes machines (cf. figure 6).
Pour le problème de tabulation, on se restreint aux fonctions polynomiales de degré n :

> « Soit f une fonction polynomiale de degré n. Calculer les images par cette fonction de nombres x_0, x_1…x_k… espacés d'un pas p et appartenant à l'intervalle [a, b] avec $x_0 = a$. »

Les différentes machines dans chacune des trois situations de l'ingénierie didactique sont analogues de celles de la genèse historique des machines : *machine Arithmétique, Machine Analytique de Babbage et enfin Machine Ordinateur de Von Neumann* (cf. figure 6).

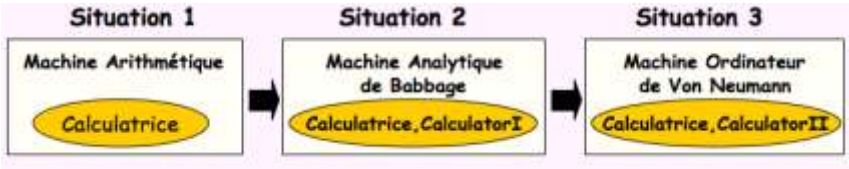


Figure 6. Les trois situations de l'ingénierie didactique

Les enjeux de chacune des 3 situations par rapport au problème informatique sont les suivants :
Situation 1 : Exploration des pratiques instrumentées avec la calculatrice et disponibilité des mémoires.
Situation 2 : Formulation de l'invariant de boucle à une machine à mémoire effaçable.
Situation 3 : Écriture d'un programme informatique à boucle à la machine ordinateur.
Nous faisons l'hypothèse que la calculatrice non programmable fonctionne pour l'élève comme une machine arithmétique *malgré la présence de mémoires effaçables et l'écriture d'un programme extérieur*. Dans les situations 2 et 3 la calculatrice est complétée par des parties fictives qui sont les robots Calculator I et II. Nous verrons plus loin les propriétés de ces robots.

2. La situation 1 (Alpro)

2.1. Description

Dans cette première situation, nous proposons trois calculs effectifs, les deux premiers individuellement, le troisième en binôme, avec écriture d'un message à un autre binôme en langage calculatrice (écriture des touches à appuyer uniquement). On peut considérer ces calculs comme le calcul de l'image d'un seul nombre $x \in Dj$ par une fonction polynomiale ou composée.
Nous allons associer à chaque calcul des stratégies optimales en termes de coût (nombre d'appuis de touches et risque d'erreurs) en rapport avec l'usage des mémoires.
- La consigne pour le calcul 1 est la suivante : « Calculer $2x^2 + x + 1$ avec x = 3,141 ».
Pour ce calcul, toutes les stratégies sont équivalentes (entre 16 et 20 appuis).
- La consigne pour le calcul 2 est la suivante :
« Calculer $8x^3 + 6x^2 - 3x - 1$ avec x = 3,141759682 »[44].
Pour ce calcul, les stratégies utilisant la mémoire Ans ou les mémoires ABC sont équivalentes (entre 29 et 37 appuis). Par contre les stratégies sans usage de mémoire sont très coûteuses (jusqu'à 134 appuis !).
- La consigne pour le calcul 3 est la suivante :

> « Ecrivez 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.
> $z = y^2 - 5y + 7$ où $y = \frac{2x^3 + 15x - 5}{32,1x - 27,7534}$ avec x = 1,254 ».

Pour le calcul 3, qui peut être interprété comme le calcul de l'image par une fonction composée, les stratégies à mémoire sont obligatoires pour obtenir le résultat du calcul.

2.2. Quelques résultats

Dans les tâches de calcul numérique instrumenté, le rapport à la calculatrice dans EMS est essentiellement celui d'une machine arithmétique, c'est-à-dire sans mémoire effaçable comme le montre le tableau 4 suivant, dans lequel le fonctionnement non intentionnel de la mémoire Ans est noté Ans~~Sto~~.

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 4. Usage des mémoires

[44] Une consigne supplémentaire est : « Combien de fois en tout devez-vous appuyer sur les touches de la calculatrice Alpro pour réaliser le calcul complet de la question 2 ? ».

(issu de l'analyse des fichiers historiques d'Alpro de 94 élèves) de 2nde et 1ère (2004-05).

L'expérimentation de la situation 1 a été conduite dans cinq classes :
- Deux classes de 2nde , France
- Une classe 10 (équivalent de la 2nde) et une classe 11 (équivalent de la 1ère), Viêt-nam
Les procédures très largement majoritaires sont des procédures de calcul sans usage de mémoire ou sans usage intentionnel de mémoire.
L'exécution du programme écrit à autrui pour le calcul 3 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 comme l'attestent les programmes en actes et les programmes écrits des élèves.
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.

3. Situations 2 et 3 (Alpro, Calculator I puis Calculator II)

3.1. Description

C'est dans ces deux situations que l'on aborde véritablement le problème mathématique de tabulation et de la répétition d'un même calcul.

> « Soit **f** la fonction $\mathbf{x \rightarrow x^2 + 1}$ Calculer les images par cette fonction de m nombres $x_0, x_1 \ldots x_k \ldots$ espacés d'un pas **p** et appartenant à l'intervalle **[-3, 2] avec $x_0 = -3$ ».**

On va jouer sur la taille du pas pour augmenter le nombre de répétitions de l'invariant de calcul.
Deux algorithmes implicites prenant en charge 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 : $x_{k+1} = a + k.p$ avec $a = -3$
Le nombre n de répétitions doit être calculé en faisant intervenir la valeur du pas et la longueur de l'intervalle.
L'écriture de ces algorithmes fait partie du travail mathématique indispensable pour rendre programmable la solution mathématique du problème de tabulation.
- Dans la situation 2, le passage de la valeur 0,2 à la valeur 0,03 pour la valeur du pas p, augmente drastiquement, non seulement la complexité du calcul (saut informationnel) mais aussi l'écriture du programme. En particulier, 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. On passe ainsi de 26 à 167 répétitions de l'invariant du calcul !
Pour 167 répétition, on doit recourir à l'aide d'un robot *Calculator I* qui joue le rôle d'une unité de commande des calculs indiqués dans un programme extérieur.
Ce robot ne sait exécuter que deux actions :

• Appuyer sur les touches de la calculatrice Alpro à condition qu'on lui indique la suite d'appuis des touches par écrit ;
• Imprimer automatiquement ce qui est affiché sur la ligne de résultat de la calculatrice Alpro après l'appui sur la touche = ;
De plus Calculator I ne possède ni papier ni stylo.
Les capacités de la machine (Alpro, Calculator I) font que la longueur du message est au moins le nombre d'appuis de touches d'Alpro. L'écriture du message est alors impossible dans un temps limité !
- Les enjeux du passage de la situation 2 à la situation 3 sont la co-amélioration de la machine et du langage pour pouvoir diminuer drastiquement la longueur du message. La transformation de l'invariant de répétition en un corps de boucle est une réponse à cette recherche d'économie. Pour cela on introduit un robot amélioré *Calculator II*, qui a la capacité supplémentaire (par rapport à Calculator I) de comprendre des instructions formées d'un nombre limité de mots (3-5 mots) exprimant la répétition (répéter, fois, Fin etc.) ; on cherche donc à faire évoluer le langage à la machine en même temps que la machine (Alpro, Calculator II) qui devient un machine à programme enregistré c'est-à-dire une machine de Von Neumann.
Pour cela, on demande aux élèves :
• 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 amélioré Calculator II devra pouvoir répéter de lui-même) ;
• 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, en complétant le langage Alpro par au plus 5 mots : grâce à ce nouveau langage, aux instructions de séquentialité, présentes dès la situation 1, s'ajoutent des instructions d'initialisation du programme, d'itération et de sortie de boucle.

3.2. Situations 2 : résultats

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 l'opération fondamentale d'affectation était problématique « même pour des élèves d'un plus haut niveau de connaissance dans la construction des boucles » (Samurçay 1985). Cependant un nombre quasi équivalent d'élèves continue à concevoir les mémoires variables A, B, C comme des variables mathématiques. Ce statut, présent dès la situation 1, résiste aux conditions mises en place, en particulier à la répétition importante des calculs. Les élèves ont recours à des palliatifs comme la mémoire papier ou des symboles ou mots habituels de la répétition.
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 I 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 I qui ne sait ni répéter ni arrêter des calculs.
L'enseignant a du intervenir pour rappeler les capacités et les limitations du robot Calculator I.
La situation 2 s'achève sur le besoin d'améliorer de robot Calculator I puisque l'écriture du programme en un temps limité est impossible !

3.2. Situations 3 : résultats

Les programmes écrits par les binômes à la fin de cette situation attestent de 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,
- 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.
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 ont permis :
- de valider ou d'invalider les 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 situation 3 a permis aussi d'observer la difficulté de la mise en place d'une variable compteur.
Maintenant, nous allons examiner de plus près le parcours d'un binôme de 1ère S en France (programme 2000).

Etude du cas d'un binôme de 1ère S en France

1. Situation 1 : d'Alpro « machine Arithmétique » à Alpro « machine à mémoires variables mathématiques »

Nous ne regardons ici que la phase du calcul 3. Rappelons ce calcul :

« $z = y^2 - 5y + 7$ où $y = \frac{2x^3 + 15x - 5}{32{,}1x - 27{,}7534}$ avec $x = 1{,}254$ ».

Chaque binôme dispose d'un stylo et d'Alpro et doit écrire un message le plus court possible à un autre binôme qui dispose aussi d'Alpro, mais qui n'a ni papier ni stylo. Dans le message, on ne doit indiquer que les touches de la calculatrice.
Cette phase de codage est suivie :
- d'une phase de décodage du message : exécution du programme du message sur Alpro. Cette exécution valide ou invalide pragmatiquement le programme. Le binôme décodeur envoie le résultat de l'exécution de son programme au binôme codeur ;
- d'une phase de rectification du programme suite à la phase de décodage du message.

Examinons le message écrit par le binôme de 1ère observé.

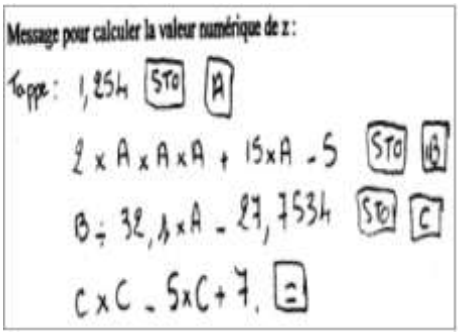


Figure 7. Message du binôme observé pour le calcul 3

Le programme écrit utilise la suite des mémoires A, B, Ans et C.
Mais les mémoires A, B, C ne sont disponibles qu'avec le statut de variable mathématique, c'est-à-dire elles ne contiennent qu'une seule valeur et elles n'utilisent pas leur propriété d'éffaçabilité.

2. Situation 2 : émergence de la notion de variable comme mémoire effaçable

Dans cette situation, on joue 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 : le nombre de répétition passe de 26 à 167
Dans la phase 1 de la situation 2, le travail mathématique du binôme observé s'appuie sur l'algorithme MF : $x_{k+1} = - 3 + k \times p$. Il utilise la mémoire Ans pour réutiliser le dernier calcul réalisé (avec un statut de variable mathématique).
L'enjeu de la phase 2 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* (déjà initié lors du calcul 3 de la situation 1) : suite de touches d'Alpro. Le travail mathématique du binôme dans cette phase s'appuie *sur* l'algorithme MR : $x_{k+1} = x_k + 0{,}03$

Examinons les trois messages successifs adressé au robot Calculator I, messages que nous qualifions de programme.
En continuité avec la phase 1 le binôme utilise la mémoire Ans avec un statut de variable mathématique avec une erreur : -1 au lieu de +1 dans le calcul de l'image (cf. figure 8).

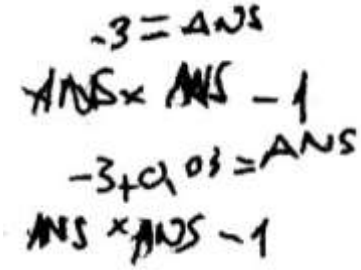


Figure 8. Premier programme avec mémoire Ans : statut de variable mathématique

L'un des élèves met en doute ce programme et propose d'utiliser une mémoire variable. Aussitôt, l'autre élève lui dicte un autre programme (cf. figure 9). Ce programme reste inachevé. L'intention est que la mémoire variable A contienne une constante, le pas p.

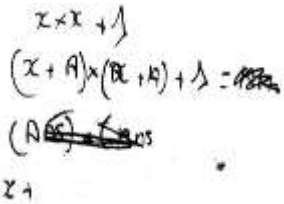


Figure 9. Deuxième programme avec mémoire A pour contenir le pas

Le stockage des nombres 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 les pointillés puis le trait qui renvoie à A + 0,03.

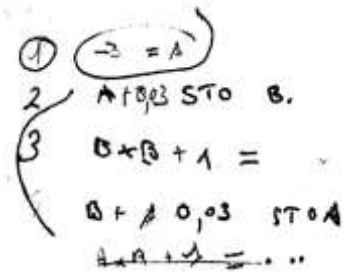


Figure 10. Troisième programme mémoires variables A, B : émergence de l'effaçabilité

L'opération effective de mise à jour « A + 0.03 Sto A » n'est pas encore disponible (utilisation d'une mémoire intermédiaire B). Mais les élèves différencient déjà de ce qui relève de l'initialisation de ce qui relève du corps de boucle. Ils ne discutent pas des capacités de Calculator I et confient donc la tâche de répéter le corps de l'invariant de la répétition à Calculator I.

3. Situation 3 : de la programmation d'une machine à mémoire effaçable à la programmation d'une machine de Von Neumann

En même temps que l'émergence de la notion de boucle, nous avons conçu le passage de la situation 2 à la situation 3, 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'affectation, d'itération et de sortie de boucle.
Dans la première phase de cette situation on demande aux élèves d'écrire le groupe de touches à répéter (cf. Figure 11).

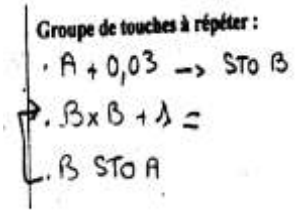


Figure 11. Groupe de touches à répéter écrit par le binôme

Notons que leur groupe de touches ne permet pas de calculer f(-3). Ils continuent à utiliser une mise à jour indirecte c'est-à-dire qui s'appuie sur une mémoire intermédiaire B. Cette difficulté à formuler cette composante essentielle du corps de boucle qu'est la mise à jour des variables informatiques est générale.
Dans la phase de synthèse le professeur institutionnalise les deux groupes de touches associés à chacun des algorithme de calcul M(R) et M(F). Nous ne donnons ici que celui retenu par le binôme observé, l'algorithme M(R)

AxA +1 =
A+0,03 Sto A

Les groupes de touches restent affichées au tableau.
Dans la troisième phase, on introduit le robot Calculator II à qui on attribue la capacité supplémentaire (par rapport au robot Claculator I) de pouvoir répéter l'un des groupes de touche écrit au tableau, à condition de choisir des mots qui permettent d'écrire un message le plus court possible (les mots sont comptés comme les touches).
Voici le message écrit à la machine (Alpro, Calculator II) par notre binôme (cf. figure 12).

Figure 12. Message final à la machine (Alpro, Claculator 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.

Conclusion

1. Les principaux résultats

- *L'effaçabilité de la mémoire et la notion de variable*

La notion de variable informatique se construit contre la notion préconstruite de variable mathématique. Elle 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 l'ingénierie didactique. Cette notion de variable a besoin pour émerger 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 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 institutionnelle et les résultats de la première situation de l'ingénierie mettent en lumière un fonctionnement largement majoritaire de la calculatrice comme une machine arithmétique : les touches mémoires peuvent être utilisées sans donner intentionnellement à la mémoire le caractère d'effaçabilité.

- *L'économie de la communication homme - machine : 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 Lovelace (1842).
Nous avons vérifié que l'intention d'exécuter des calculs répétitifs avec un invariant opératoire est une condition 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.
- *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 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.

2. Les limites de ce travail

- 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. Il faudrait 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.
- 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 par le tableur dans EMS en France.
Il semble judicieux d'envisager une reprise de l'ingénierie didactique sur un problème mathématique du domaine de l'Arithmétique ou de la Statistique.
-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 des deux pays pour l'introduction de ces objets, pour l'utilisation de la calculatrice et de l'ordinateur.

Bibliographie

Bouvier A., George M. Le Lyonnais F. (1993) *Dictionnaire de mathématiques*. Paris : PUF.

Golschlager et Lister (1986) *Informatique et algorithmique*. InterEdition.

Horowitz E. et Sahni S. (1978) *Fondamentals of Computer Algorithms*. Edition Pitman.

Knuth D-E. (1968) *Algorithmes fondamentaux, l'art de la programmation d'ordinateur*. Addison-Wesley Publishing Company.

Kuntzmann J. (1957) *Formation de calculateurs – Moyens de Calcul – L'atelier Arithmétique*. 2e édition. Université de Grenoble, Année 1957 – 1958.

Lê Van T. (2001) *Etude didactique de liens entre fonctions et équations dans l'enseignement des mathématiques au lycée en France et au Viêt-nam.* Thèse en cotutelle France – Viêt-nam de didactique des mathématiques. Université Joseph Fourier (Grenoble) et Université Pédagogique de Ho Chi Minh.

Ligonnière R. (1987) *Préhistoire et histoire des ordinateurs*. Paris : Edition Robert Laffont.

Nguyen C. T. (2005) *Étude didactique de l'introduction d'éléments d'algorithmique et de programmation dans l'enseignement mathématique secondaire à l'aide de la calculatrice.* Thèse en cotutelle France – Viêt-nam de didactique des mathématiques. Université Joseph Fourier (Grenoble) et l'École Normale Supérieure n°1 de Hanoi.

Samurçay R. (1985) Signification et fonctionnement du concept de variable informatique chez des élèves débutants. *Educationnal Studies in Mathematic,* n° 16-2 (pp. 143-161).

Swade D. (1993) Le calculateur mécanique de Charles Babbage. *Pour la Science*, n° 186, (pp. 78-84).

Verroust G. (2004) *Histoire, épistémologie de l'informatique et révolution technologique*. http://hypermedia.univ-paris8.fr/Verroust/cours/

Dictionnaire Encylopaedia, tomes 1, 12, 19

Rapport de la commission Kahane (2001)

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 2e, 1er S, Edition Belin 2000

2. Collection Déclic Maths 2e, 1er S, Tle S, Edition Hachette 2001

3. Collection Mathématiques 2e, 1er S, Tle 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