Algorithmique sans problème ?

Jean-Jacques Dhénin(*)

1 Posons le problème (le temps de voir)

Notre génération se trouve mise en présence aujourd'hui avec l'enseignement de l'algorithmique. Pour l'heure, les opinions divergent et toutes les directions ne se valent pas [Dhénin1]. Comme pour enseigner les mathématiques, cela suppose une réflexion sur les objectifs de la discipline [Bkouche].

Essayons de comprendre les questions. Nous en connaissons vraisemblablement un *bout* puisque nous utilisons déjà le mot, *algorithme*(1).

D'autre part, la lecture des divers documents mis à notre disposition nous renseigne sur la commande sociale [Eduscol]. Ce qui est frappant dans cette prescription, c'est le fait que l'enseignement de l'algorithmique n'est pas indiqué comme transmission de connaissances mais, au mieux, comme une pratique. Pourtant, l'avenir ne devrait pas se réduire à un aménagement du présent et prévoir certaines difficultés pourrait nous épargner bien des tracas.

Qu'est-ce qui est vain ? Apprendre sans réfléchir ou réfléchir sans apprendre ? Considérons plutôt que notre métier, entre autres, **consiste à transmettre des connaissances** aux élèves. Cela ne signifie ni un désintérêt pour les élèves, ni un dédain pour la pédagogie. C'est bien plutôt que le sens, en pédagogie, commence en amont, il est intrinsèque à la discipline elle-même.

Langage : maîtrise ou domination ?

Donnons au problème une forme telle qu'il soit possible de le résoudre [Vuillemin],

Certains obstacles en mathématiques, pour nos élèves, sont en fait des difficultés de français : ils *comprennent* mal nos énoncés et notre enseignement. C'est ce que dit depuis fort longtemps Stella Baruk [Leguay] ; nous devrions être toujours prudents

(*) dhenin@gmail.com

(1) Pour, par exemple, décrire les différentes étapes de la division ou, plus ouvertement, en exposant aux élèves le mythique *Algorithme d'Euclide*. En revanche, on ne parle pas d'algorithme pour le calcul du périmètre d'un cercle. Nous ne confondons pas « algorithme » et « programme », nous distinguons bien « calcul » et recherche au moyen d'une « méthode » ou d'un « mode opératoire » qui évite le tâtonnement.

de nos mots et de tout ce qu'ils portent comme sens usuel[2].

L'algorithmique apporte un nouveau vocabulaire qui nécessite d'autant plus d'explications qu'il n'est pas d'usage quotidien pour la majorité des élèves. Les mots *entrer*, *réitérer*, *valeur de n*, *tester*, *itération*, *étape*, *modulo*, *initialiser*, *remplacer*, *expliciter*, *commentaires*, *instructions*, *présupposition*, … peuvent sembler avoir un sens évident, pour qui pratique la programmation, pas pour le débutant[3]. Nous prendrons soin de n'en faire usage qu'après nous être assurés d'être compris. Éclairer n'est pas éblouir.

Considérons, par exemple, le mot « affectation » utilisé par la programmation impérative, la plus répandue (Fortran, Pascal, C, …) et la plus proche de la machine réelle. Son principe est de modifier à chaque instant l'état de la mémoire via les affectations.

- Ce mode de programmation présente plusieurs défauts pour un lycéen quelle que soit l'écriture retenue[4].
  Dès lors, on ne peut plus parler de la valeur de la variable x. Il faut parler de la suite des valeurs que prend cette variable, et l'on ne peut en rendre compte correctement qu'en reprenant le mécanisme exact d'exécution du programme.
  Ce n'est pas là le seul inconvénient de l'affectation ; on s'éloigne des mathématiques en utilisant l'affectation comme base de programmation : un élève de seconde découvre les fonctions qui, à un nombre donné, associent une unique image, comment comprendra-t-il qu'une même variable peut prendre des valeurs tout à fait différentes selon le moment où on l'évalue ?

« Un instant, svp !
Hier vous disiez que x valait 2 ! »

Une expression mathématique, telle que $y = 5x + 3$ sous-entend que la valeur de y dépend de celle de x. Si nous connaissons x nous pouvons connaître y. De même en algorithmique si $y \leftarrow 5x + 3$ ne présente pas de difficulté pour moi. pour l'élève la difficulté réside dans l'inversion du sens de la flèche. La flèche n'a plus le *même sens*. Que dire de $x \mapsto 5x + 3$ et de $x \leftarrow 5x + 3$? Bien sur si l'on ne se fonde que sur les élèves « mordus d'informatique » on aura l'illusion d'être compris ; mais chacun sait…

(2) Un exemple : depuis que nous avons évacué le mot « pente » de notre vocabulaire concernant les droites, nous constatons que les élèves ne comprennent pas que nous déclarions qu'une fonction affine est croissante et que « ça se voit »... Ils ne perçoivent pas le sens de lecture de la gauche vers la droite ; le mot « pente », utilisé par de nombreux francophones, induisait comment il convient de « voir ».

(3) Combien d'élèves, surtout au collège, s'approprient les termes propres à notre discipline (les mathématiques) qu'ils utilisent ensuite tellement maladroitement qu'ils semblent n'y rien comprendre.

(4) $x \leftarrow 5$ ou $x := 5$ ou encore $x = 5$ ou même $5 \longrightarrow L_1$ pour une calculatrice TI82.
Une part importante de notre travail d'enseignant gravite autour de la notion d'égalité. (École républicaine oblige !). Un coup d'œil sur le sujet récent « Égalité des segments » dans la liste maths-collège est édifiant : 35 interventions en 3 ou 4 jours.

- une variable peut être modifiée en plusieurs endroits du programme, et cela est vraiment une grande source de difficultés dans la programmation impérative.

Mieux vaut allumer une bougie que maudire les ténèbres [Lao Tseu].

Or il y a trois modes de programmation : la programmation constructive (ou impérative, celle qui utilise l'instruction d'affectation), la construction implicite (ou récursive, ou fonctionnelle, ou explicative, …) et la programmation logique. Ce sont trois modes de pensée différents. Pour chaque mode, il y a différentes méthodes pour élaborer un programme.

L'importation de ces méthodes (produites par les pratiques professionnelles) dans des situations d'enseignement est délicate. Une part de la difficulté réside dans le choix des exemples. Le problème choisi doit être assez complexe pour que la méthode proposée soit acceptée comme un outil pertinent et, pour que l'outil soit opérant avec des élèves dont les acquisitions sont encore en cours de maturation, il faut que la complexité du problème soit « gérable » dans les conditions de travail de la classe ; attendons-nous à voir refleurir les recettes qui ont fait leur preuves : réorganisation de lignes volontairement mélangées, exercice à trous (*j'ai 2 ailes et un moteur*, …, je suis … … **content**) . et espérons que nous saurons faire preuve d'innovation !

2. Soulevons le problème (le temps de comprendre)

Consacrons un moment à la recherche d'un algorithme. Cet examen sera riche d'enseignement.

« Au moyen de deux récipients dont les capacités respectives sont neuf litres et quatre litres, nous souhaitons disposer d'une quantité d'eau de six litres ».

On retrouve ici les mêmes ingrédients que dans la recherche d'un algorithme : on dispose d'une situation initiale qu'il faut préciser (deux récipients vides), d'un résultat recherché (6 l dans le grand récipient), et d'un petit nombre d'opérations possibles (remplir un récipient au robinet, vider un récipient dans l'autre ou dans un évier) ; il convient de construire une liste ordonnée de ces opérations qui conduise à coup sûr au résultat.

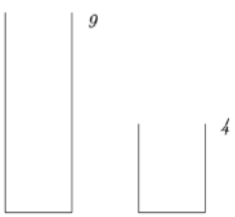


Figure 1

Représentons-nous les deux récipients. Du fait de l'absence de graduation, nous ne savons pas comment mesurer exactement ; tâtonnons un peu. *Le tâtonnement est la forme la plus primitive de recherche d'une solution.* Nous pouvons remplir complètement le plus grand ; si, avec son contenu, nous remplissons alors le petit, il nous reste cinq litres dans le grand. Pouvons-nous également en obtenir six? Vidons à nouveau les deux récipients. Nous pourrions aussi…

Nous agissons ainsi comme la plupart des gens à qui l'on pose ce problème. Nous faisons un essai, puis un autre, et, après chaque échec, nous recommençons et

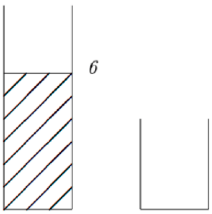


Figure 2

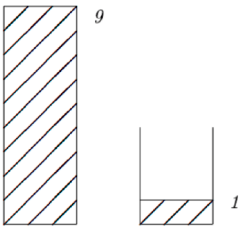


Figure 3

cherchons autre chose. En somme, nous progressons, en partant de la situation donnée au début, vers la situation finale désirée, c'est à dire en allant du connu vers l'inconnu. Il se peut qu'après maintes tentatives nous finissions par réussir, mais ce sera par hasard.

Que nous demande-t-on ? Représentons-nous la situation finale comme à la figure 2 (*partons de ce qui est demandé et admettons que ce que l'on cherche est déjà trouvé*). À partir de quelle situation précédente pourrions-nous obtenir la situation désirée (*cherchons à partir de quel antécédent le résultat final pourrait être obtenu*). Le grand récipient rempli, il faudrait en retirer trois litres exactement. Pour cela … il faudrait **avoir déjà un litre dans le petit**. Voilà l'idée ! (Cf. fig. 3).

Cherchons à nouveau quel pourrait être l'antécédent de cet antécédent. La situation de la figure 2 est équivalente aux situations des figures 3 et 4. Si l'on obtient l'une quelconque des situations des figures 2, 3 et 4, on obtiendra aussi bien les deux autres.

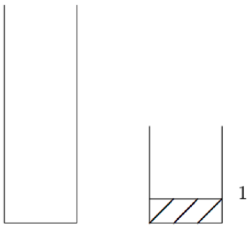


Figure 4

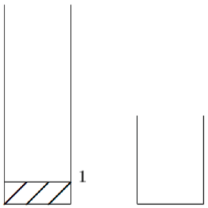


Figure 5

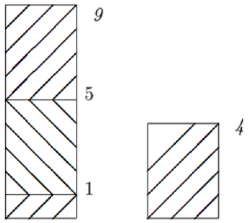
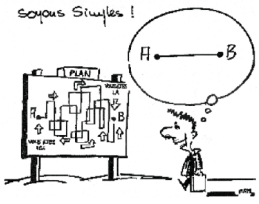


Figure 6

Ainsi, par *raisonnement régressif* nous avons découvert la succession d'opérations appropriées. Nous n'avons plus qu'à *renverser le processus en partant du dernier point atteint dans notre analyse*. Nous faisons les opérations suggérées par la figure 5 et obtenons la figure 4, puis nous passons à la figure 3, de là à la figure 2 et finalement à la figure 1.



Il y a certainement quelque chose d'assez profond dans cette méthode. L'obligation de s'éloigner du but, de ne pas prendre la route qui mène directement au résultat désiré, pour découvrir la succession d'opérations appropriées, suit **un ordre exactement à l'inverse de l'ordre réel**. Il n'est nul besoin de génie pour résoudre un problème en revenant en arrière.

Le but n'est pas le but, c'est la voie [Lao Tseu]

Chacun d'entre nous peut vérifier que les élèves en difficulté disent souvent : « Je ne sais pas *comment* faire! » et qu'il nous est facile, alors, de demander : « Faire *quoi* ? » tant nous savons bien que c'est l'objectif qui détermine, en grande partie, le parcours. Cela est vrai pour l'algorithmique mais aussi pour résoudre les problèmes généralement… (excepté, peut-être, les problèmes financiers).

Utilisons cet acquis pour analyser une autre situation simple qui met en jeu également l'affectation. L'affectation sera la principale difficulté que nous aurons à éclaircir avec les élèves.

3. Reposons le problème (le temps d'une première conclusion)

La forme, c'est le fond qui fait surface [Victor Hugo]

J'ai essayé de systématiser la méthode précédente au moyen d'une table d'analyse, c'est le premier outil que les élèves acquerront.

Cette table d'analyse (*figure 7*) intervient lors de l'étape de l'écriture de l'algorithme avec un métalangage. Elle permet d'organiser méthodiquement l'expression de l'algorithme. C'est un tableau de trois colonnes réalisant le schéma dans lequel l'algorithme s'organise et se développe. Les élèves commencent par placer en haut de la colonne centrale la **dernière** action, ce qui fait apparaître, au moins, une variable que l'on va chercher à expliciter. Cette variable en présuppose d'autres que l'on explicite, et ainsi de suite jusqu'à ce que toutes les variables soient expliquées ou explicitées. Dans la colonne *Lexique*, on précise pour chaque variable son *domaine de définition*. Les élèves terminent en fixant dans la colonne de droite l'ordre d'exécution des séquences pour le programme. Ainsi que le souligne Gérard Kuntz dans ses formations de « l'option informatique des lycées », cette table est constituée de trois parties : les définitions lexicales, les définitions formelles (algorithmiques), l'ordre des définitions qui définit l'algorithme. Vous remarquerez que les définitions formelles constituent exactement un « système d'équations » (au sens mathématique) qui permet de calculer toutes les inconnues (à partir des données et des définitions). Cette remarque est tout à fait générale et fait le lien entre algorithmique et mathématiques.

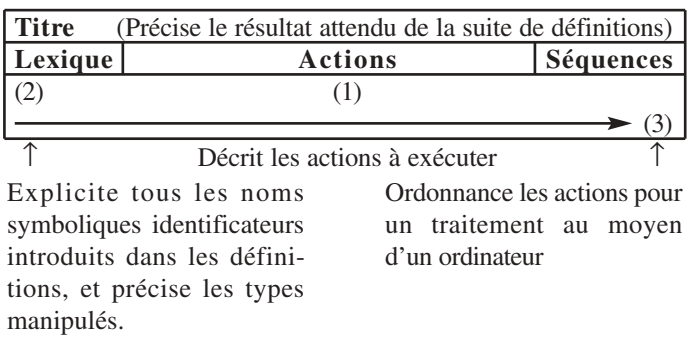


Figure 7

« Soit par exemple à fabriquer une série d'appareils dont chaque exemplaire sera équipé d'un quartz et de deux diviseurs de fréquence correspondant à la demande de chaque client. À la commande, le client précisera les deux fréquences dont il a besoin. Nous allons écrire le programme qui détermine la valeur du quartz et les valeurs des deux diviseurs de fréquence. sachant que toutes ces valeurs sont des nombres entiers ».

Il n'est pas nécessaire d'être ingénieur pour comprendre que le quartz sera un multiple des fréquences souhaitées. On choisira le plus petit multiple. En revanche, nos connaissances mathématiques nous fournissent le calcul de ce plus petit commun multiple au moyen du PGCD.

FREQ : /* Calcul d'une fréquence et de deux diviseurs */		
Lexique	Actions	ordre
F_q : Fréquence du quartz	$F_q \leftarrow$ PPCM(F_1,F_2) ;	2
F_1 et F_2 : Fréquences client	$div_1 \leftarrow \frac{F_q}{F_1}$;	3
	$div_2 \leftarrow \frac{F_q}{F_2}$;	4
div_1 : diviseur pour F_1	SAISIR(F_1,F_2) ;	1
div_2 : diviseur pour F_2	AFFICHER(F_q, div_1, div_2) ;	5
PPCM($x \downarrow$, $y \downarrow$) /* Calcul du Plus Petit Commun Multiple */		
	PPCM $\leftarrow \frac{x \cdot y}{\text{PGCD}(x,y)}$	
PGCD($x \downarrow$, $y \downarrow$) /* Calcul du Plus Grand Commun Diviseur */		
m : variable temporaire	TANT QUE $x \neq 0$ $m := x$; $x := y$; $y := m$ MOD y ; QT PGCD $\leftarrow y$;	

Figure 8. Table d'analyse des calculs des fréquences

4. Déplaçons le problème (le temps d'une deuxième conclusion)

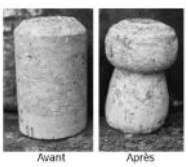
L'algorithmique n'est ni une science expérimentale, ni un gadget pédagogique.

Nous disposons maintenant d'un guide de construction pour nos premiers algorithmes. En revanche, il nous manque un **outil de validation**. Nous en avons fini avec l'analyse. Pour se rendre compte de la difficulté, on peut consulter de nombreux exercices proposés sur Internet[5].

(5) Par http://www.vincentobaton.fr/MathsLycee/DocsSeconde/2009_2010/Seconde

Viendrait-il à l'idée de commencer l'étude de la résolution des équations sans, au préalable, avoir fait travailler les élèves sur la priorité des opérations ? De même, est-il prudent de faire précéder la construction d'algorithme par *l'apprentissage de la lecture, de la compréhension, et de l'écriture* de la suite des actions.

Un programme décrit la suite de transformations qui fait passer de la situation initiale à la situation finale, en passant par toute une suite de situations intermédiaires. Les situations peuvent être décrites au moyen de *prédicats reliant les variables* du programme. On va donc isoler dans la suite des situations quelques-unes d'entre elles considérées comme les « pivots » du programme, puis dire comment on passe de l'une à l'autre. Pour cela il convient de s'habituer à placer de vrais « commentaires » entre les actions, pas des paraphrases comme « // J'augmente x de 1 » qui n'apporte rien à la compréhension.



L'algorithme fait intervenir la notion de temps. « Avant » l'instruction c'est la pré-assertion. « après » c'est la post-assertion. Ces assertions décrivent des états (relation entre les variables), les instructions font passer d'un état à l'autre.

Savoir donner du sens

Sachant où l'on est et le but à atteindre, on sait quel chemin prendre.

- Commençons avec deux exemples simples et voyons comment les commentaires qui encadrent l'affectation nous renseignent sur le sens de l'action.

  – Si $a = x^{i+1}$ et si l'on effectue $a = a \times x$, on pourra écrire :

// $a = x^{i+1}$	← Situation tenue pour vraie ;
a PRENDLAVALEUR $a \times x$	← la deuxième ligne indique l'action ;
// $a = x^{i+2}$	← la troisième exprime la nouvelle relation.

  – Supposons maintenant que, sans modifier ni x, ni a, nous ayons besoin d'avoir a écrit sous la forme x^{i+1} et que le travail déjà accompli soit tel que $a = x^{i+2}$; nous devons écrire $i = i + 1$, ce qui demande un instant de réflexion (ce travail de réflexion, mené avec une classe, à l'occasion de l'étude du programme suivant, est toujours un moment savoureux !) :

// $a = x^{i+2}$	← Situation de départ ;
i PRENDLAVALEUR $i + 1$	← action :
// $a = x^{i+1}$	← la relation que l'on veut atteindre.

- Le déroulement linéaire, ou enchaînement, est une succession d'instructions pour se rapprocher d'un **résultat prévu** ; les relations entre les variables sont

modifiées. Bien entendu l'état final est le résultat de la succession des états intermédiaires.

```
// a = x^(i+1)
a PRENDLAVALEUR a × x
i PRENDLAVALEUR i + 1
// a = x^(i+1)
```

On utilisera les deux exemples précédents pour interpréter cet enchaînement.

M'sieu, j'ai bon ? M'sieu,

Les prescripteurs laissent entendre que l'épreuve de la programmation, sorte d'épreuve du réel, pourrait servir à la validation impartiale du travail de l'élève. Certes l'idée est répandue, voire même séduisante ; mais c'est un leurre. Jacques Arsac [Arsac1] en a fait une illustration remarquable dont je me suis inspiré pour montrer aux élèves qu'un programme qui fonctionne n'est pas nécessairement un algorithme correct [Dhénin3]. La principale difficulté est d'éviter de travailler à tâtons, « On error try again » ; il convient de donner, dès que possible, du sens à ce que l'on écrit.

L'exemple ci-dessous(6) en constitue une preuve irréfutable. *Essayer un programme*, avec succès ne prouve pas qu'il soit juste. C'est la raison pour laquelle je tenais à l'examiner afin de montrer « comment chasser l'erreur ».

```
Lire x
Lire n
a PREND_LA_VALEUR x
i PREND_LA_VALEUR 0
t PREND_LA_VALEUR n
   TANT_QUE (i < t) FAIRE
      a PREND_LA_VALEUR a × x
      i PREND_LA_VALEUR i + 1
      t PREND_LA_VALEUR n − 1
   FIN_TANT_QUE
AFFICHER a
```

Essayons ce programme avec $x = 5$ et $n = 3$; réponse 125. Essayons encore avec $x = 4$ et $n = 2$; réponse 16. La première ligne de TANT_QUE $a = \times x$ semble bien montrer que l'on calcule x^n par multiplications successives. Nous pouvons être satisfaits et nous arrêter là.

Pourtant, un dernier essai avec $x = 10$ et $n = 1$ s'avère poser problème. Un examen détaillé du déroulement de l'algorithme pour $n = 1$ et pour $n = 2$ révélera aux élèves le pourquoi de cette difficulté.

Construire une boucle

Dès que le programme comporte une boucle, les choses se compliquent singulièrement.

On peut écrire l'algorithme en utilisant le tableau d'analyse vu dans la première partie :

Résultat – écrire « x puissance n = », a_f (a final)	4
a_f = pour i allant de 0 à n répéter a = @$a \times x$ (a courant = (ancien a) $\times x$)	3
a_0 = 1 ((a origine) = 1)	2
x,n = données	1

(6) J'utilise AlgoBox [Brachet] parce qu'il est le plus près de ce que l'on peut demander aux élèves de seconde et par ce qu'il offre de possibilités de partage Internet.

On met ainsi en évidence une situation absolument générale : toute itération (simple ou générale) produit essentiellement des *suites* au sens mathématique du terme, avec le principe suivant :

À l'issue de l'itération, on obtient une valeur *finale*.

Le module itéré (ici une instruction unique) fait passer d'une ancienne valeur de l'identificateur à la valeur suivante (la valeur courante).

Pour que l'itération soit possible, il faut initialiser l'identificateur (valeur origine).

Les quatre occurrences de a (notées de quatre façons différentes dans la table : a_0, @a, a, a_f) vont se fondre en un a unique dans le programme : on voit bien la difficulté et l'utilité d'analyser avant de programmer).

Avec ce formalisme, on vérifie d'un coup d'œil si l'ensemble tient la route.

Il devient très difficile de « penser » le déroulement, notamment la condition d'arrêt [Dhénin2]. Il n'est pas facile de corriger le programme ci-dessus. Il est probable que l'auteur a mal choisi son test d'arrêt et qu'il a cherché à redresser la situation en introduisant la variable t. Son calcul dans la boucle est inattendu. Les deux exemples du § précédent s'appliquent ici pour donner l'interprétation.

Pour éviter de passer un temps considérable à rechercher une erreur d'algorithme, il est préférable d'apprendre la méthode de construction. Supposons que l'on ait fait une partie du travail et calculé $a = x^i$ pour $i \leq n$. Si $i = n$, c'est fini ; sinon, il faut se rapprocher de la solution en faisant croître i par $a = a \times x$ et $i = i + 1$. Reste à voir le démarrage, c'est-à-dire trouver des valeurs de a et i telles que l'assertion $a = x^i$ soit vérifiée pour n positif ou nul. Nous prendrons $i = 0$ et $a = x^0 = 1$; d'où le nouveau programme :

```
Lire x
Lire n
a PREND_LA_VALEUR 1
i PREND_LA_VALEUR 0
   TANT QUE (i < n) FAIRE
// a = x^i
      a PREND_LA_VALEUR a × x
// a = x^i + 1
      i PREND_LA_VALEUR i + 1
// a = x^i
   FIN_TANT_QUE
AFFICHER a
```

Proposer une situation générale : $a = x^i$.

Voir si c'est fini : $i = n$.

Progresser vers la solution et rétablir la situation générale : $a = a \times x$ et $i = i + 1$.

Démarrer le processus :
$i = 0$ et $a = x^0 = 1$.

Interpréter le résultat.

Examiner le programme pour en chasser les calculs inutiles ; on pourra, ici, supprimer le calcul de t.

Cet exemple appelle quelques remarques :

- **Les commentaires** sont nécessaires à la compréhension non seulement pour la relecture, mais même au moment de la rédaction.

- **La mise au point et la recherche d'erreur** ne se font pas quand tout est écrit. Ils se font au fur et à mesure, comme dans les devoirs de math.
- **Un algorithme** n'est pas écrit pour être lu par un ordinateur, mais par un humain ; il doit être **clair** et **simple**.

5. Ce n'est qu'un début…

Beaucoup d'entre nous se demandent : « Que peut m'apporter ce nouvel enseignement ? ». Pour ma part, j'ai trouvé une première réponse en disant que la recherche de résolution d'un problème de math peut être renforcée par l'apprentissage de la méthode de construction d'un algorithme. Identifier ce que l'on cherche[7] et remonter jusqu'aux données, nommer les objets intermédiaires intéressants, examiner comment ils sont définis, comment on les calcule, organiser les étapes d'une démarche, … Les élèves nous voient exposer l'enchaînement logique des propositions mathématiques souvent sans savoir ce que nous avons derrière la tête. De ce fait. rares sont ceux qui « devinent » d'eux-mêmes « comment faire ».

En ce qui concerne l'algorithmique elle-même, son apprentissage ne se résume pas au b-a-ba. Comme en en probabilités ou en statistique, assortir son discours d'un vocabulaire inhérent à la matière ne suffit pas : il y a un mode de penser à s'approprier. L'exposé hypothético-déductif qui fait suite à l'exclamation « Ha, ça y est, j'ai trouvé ! » est long à métaboliser. On ne se contentera pas de programmer. Ce n'est pas cela l'algorithmique.

L'algorithmique, elle aussi, nous renvoie à la question du sens. Sur ce sujet, difficile, elle illustre l'idée que **c'est l'acte de traduire qui donne un sens**. Traduire une équation de fonction par un dessin, traduire un problème par un système d'égalités ou traduire un algorithme par un programme, il est toujours question de traduire pour comprendre. L'usage des assertions est un pas dans cet apprentissage.

Références

[Arsac1] Jacques Arsac, *Les bases de la programmation*. Bordas, Paris, 1983.

[Arsac2] Jacques Arsac, *Préceptes pour programmer*. Dunod, Paris, 1991.

[Bkouche] Rudolph Bkouche, *Des tice dans l'enseignement des mathématiques*. octobre 2009, http://wwvv.univ-irem.fr/spip.php?article282.

[Brachet] Pascal Brachet (l'auteur est professeur de mathématiques au lycée Bernard Palissy à Agen), *AlgoBox 2009/2010*.
http://www.xm1math.net/algobox/index.html

[Dhénin1] Jean-Jacques Dhénin, *Carte mentale*.
http: algotaf.free.fr/Maps/AlgoSeconde.html

[Dhénin2] Jean-Jacques Dhénin, *Un exemple de boucle erronée en AlgoBox*. 2009
http://algotaf.free.fr/Code_Faux.html

[Dhénin3] Jean-Jacques Dhénin, *Correction d'un programme faux*.
http://algotaf.free.fr/tri_faux.correction.pdf

(7) La mère de Toto a trois enfants Pim Pam et ... ?

Suite page 234

[Eduscol] http://eduscol.education.fr/D0015/Doc_ress_algo_v25.pdf

[Kernigham] KERNIGHAM and PIKE, *The Practice of Programming*. Lucent Technologies, USA, 1999.

[Khawarizmi] http://fr.wikipedia.org/wiki/Al-Khawarizmi

[Lao Tseu] http://chroniquestaoistes.free.fr/html/lao_tseu001.html

[Ledgard] H. F. LEDGARD, *Proverbes de programmation*. Bordas, Paris, 1978.

[Leguay] Olivier LEGUAY, *Les maths selon Stella Baruck*, octobre 2008.
http://www.paperblog.fr/1156875/les-maths-selon-stella-baruck/

[Mathadoc]
http://www.mathadoc.com/Documents/college/3eme/3arithm/activ5.pdf

[Sesamath] http://revue.sesamath.net/spip.php?article216

[Vuillemin] Jules VUILLEMIN, La philosophie de l'Algèbre. P.U.F., Paris, 1962
http://www.univ-irem.fr/commissions/epistemologie/ressouces/ress.ext/themes/algebre.htm