Quelques algorithmes en arithmétique

Division euclidienne

- Définition
  Effectuer la division euclidienne de a par b (où a et b sont des entiers naturels, $b > 0$) signifie écrire $a = bq + r$, avec $q = E\left(\frac{a}{b}\right)$ et $r = a - bq$.
  Remarque : cette définition n'apparaît pas dans le programme de première, mais l'algorithme ci-dessous peut néanmoins être mis en place, et est utile dès la première.

- Calculatrice : programme DIV

Entrée	Entrer a et b	Prompt A,B
Traitement	Calculer et stocker q Calculer et stocker r	Int(A/B) → Q A – BQ → R
Sortie	Afficher le quotient q et le reste r	Disp Q,R

- Tableur
  Construire une feuille de calcul permettant d'effectuer une division euclidienne, selon le schéma suivant :

	A	B	C	D
1	A	B	Q	R
2	347	18	19	5

Changement de base

- Passage de la base 10 à la base b ($1 < b < 10$)
  Il s'agit d'écrire un nombre entier naturel n sous la forme : $n = a_n \times b^n + ... + a_1 \times b + a_0$.

- Calculatrice : programme BASE
  Ce programme affiche successivement les chiffres de l'écriture du nombre n dans la base b.

Entrée	Entrer b et n Stocker n dans la mémoire A	Input « BASE », B Input « NOMBRE », N N → A
Traitement	Tant que A > 0 Effectuer la division euclidienne de A par b : A = $bq + r$ Donner à A la valeur q Afficher r	While A > 0 Int(A/B) → Q A – BQ →R Q →A Pause R End
Sortie	Effectuée au fur et à mesure	

- Tableur
  Construire une feuille de calcul permettant d'effectuer un changement de base, selon le schéma suivant :

	A	B	C	D
1	Nombre N en base 10	765	Base b =	6
2	A	Q	R	
3	765	127	3	
4	127	21	1	
5	21	3	3	
6	3	0	3	

Multiplication russe

- Calculatrice : programme RUSSE

Entrée	Stocker x dans A et y dans B Stocker 0 dans M (*)	Prompt A, B $0 \rightarrow$ M
Traitement	Tant que A > 0 faire (*) Si A est impair, donner à M la valeur M+B Multiplier le contenu de B par 2 Donner à A la valeur $E\left(\frac{A}{2}\right)$	While A > 0 If fPart(A/2) > 0 M + B $\rightarrow$ M 2B $\rightarrow$ B Int(A/2) $\rightarrow$ A End
Sortie	Afficher le produit M de x et y (*)	Disp M

- Tableur
  Construire une feuille de calcul permettant d'effectuer une multiplication russe, selon le schéma suivant :

	A	B	C
1	A	B	M
2	5	2	0
3	2	4	2
4	1	8	2
5	0	16	10

Liste des diviseurs

Calculatrice : programme DIVISEURS
Ce programme affiche successivement tous les diviseurs du nombre n.

Entrée	Entrer n Stocker 1 dans d	Prompt N $1 \rightarrow$ D
Traitement	Tant que $d \leqslant \sqrt{n}$ Si d divise n, afficher d et $\frac{n}{d}$ Augmenter d de 1	While D $\leqslant \sqrt{}$ (N) If fPart (N/D) = 0 Disp D, N/D Pause D + 1 $\rightarrow$ D End
Sortie	Effectuée au fur et à mesure	

Algorithme d'Euclide

- Calculatrice : programme PGCD

Entrée	Entrer a et b ($a > b$) Calculer et stocker le reste r de la division euclidienne de a par b	Prompt A, B A – Int(A/B) × B $\rightarrow$ R
Traitement	Tant que $r > 0$ Calculer et stocker le reste r de la division euclidienne de a par b Donner à a la valeur b et à b la valeur r	While R > 0 A – Int(A/B) × B $\rightarrow$ R B $\rightarrow$ A : R $\rightarrow$ B End
Sortie	Afficher b, le PGCD	Disp B

- Tableur
  Construire une feuille de calcul permettant de calculer le PGCD de deux entiers, selon le schéma suivant :

	A	B	C
1	A	B	R
2	627	18	15
3	18	15	3
4	15	3	0

Un exercice donné en évaluation

On envisage l'algorithme suivant, utilisable pour un entier naturel non nul N :
Initialisation : Le nombre p est nul
Traitement : Pour tout entier k compris entre 1 et N,
- on effectue la division euclidienne de N par k.
- si le reste est nul, alors on ajoute 1 à p, sinon, on passe à l'entier suivant.

Sortie : On affiche p

1. Faire fonctionner cet algorithme pour N = 6 et donner le résultat obtenu pour p.
2. Que représente p dans le cas général ?

Mise en oeuvre avec les élèves

Mise en place progressive dans le temps et la difficulté :

1. Savoir éditer et exécuter un programme sur calculatrice.
   Connaître les instructions élémentaires d'entrée et sortie.
   Savoir gérer le contenu des mémoires.
   Ex : division euclidienne

2. Savoir décrire un algorithme en langage naturel
   (Phase d'analyse d'un processus et d'oralisation, permettant de mieux en comprendre les rouages. Cette étape est l'occasion de faire apparaître la notion de test et de boucle.)
   Ex : division

3. Mettre en place un algorithme permettant de répondre à un problème :
   - en langage naturel
   - sur calculatrice, connaître les tests de branchement
   - sur tableur, élaborer une feuille de calcul (sans boucle ni test d'arrêt)
   - observer le fonctionnement d'une macro faite avec OpenOffice
   (voir fichier MacroOpenOffice.pdf)
   ou d'un algorithme à l'aide du logiciel Execalgo.exe téléchargeable sur la liste de diffusion de L
   Ex : changement de base

4. Comprendre ce qu'un algorithme plus complexe produit
   Ex : multiplication russe
   Cette étape est l'occasion d'établir le lien avec la logique. En effet, l'élève est amené à imaginer et suivre le déroulement d'une suite d'instructions en fonction de conditions logiques.
   D'autre part, on peut initier les élèves à la notion de preuve d'un programme et introduire ainsi le raisonnement par récurrence.

Complément : Preuve d'un programme

En programmation impérative, il est important de prouver deux choses :
- la terminaison du programme

Prouver que le programme se termine en un nombre fini d'étapes.
- la validité du programme

On utilise un invariant de boucle pour prouver ce type de programme.
Exemple de questions à propos de l'algorithme de multiplication russe
1°) Exécuter cet algorithme pour $x = 5$ et $y = 2$, puis pour $x = 45$ et $y = 19$.
On remplira un tableau donnant pour chaque étape les valeurs prises par a, b et m.
A l'aide de ces deux exemples, conjecturer la nature du résultat m renvoyé par cet algorithme.
2°) Prouver que l'égalité $x \times y = a \times b + m$ est toujours vérifiée au niveau des (*).
En déduire que le résultat conjecturé au 1°) est bien vérifié.
Bilan : Pour prouver un programme en utilisant un invariant de boucle, il faut :
- Définir une pré-condition qui décrit l'état des variables avant d'entrer dans la boucle.
- Prouver que l'invariant de boucle est vrai à chaque passage dans la boucle.
- La condition de sortie de boucle, en conjonction avec l'invariant de boucle, permet de prouver une post-condition, ce qui valide le programme.