THÈME 18

*Depuis l'antiquité, un **algorithme** désigne une suite de calculs ou d'actions aboutissant à un certain objectif. Et depuis quelques décennies, les programmes des ordinateurs décrivent ces suites d'actions qui résolvent les problèmes de toutes sortes que se posent les femmes et les hommes d'aujourd'hui.*
*L'étude et la production de ces programmes s'appellent l'**algorithmique**…*

Algorithmique

Le mot *algorithme* vient du nom du mathématicien de langue arabe **al-Khwârizmî** (780-850), en passant par le latin du Moyen Âge.

Le plus vieil algorithme du monde…

… est certainement l'***algorithme d'Euclide***, décrit au livre 7 des *Éléments d'Euclide* : c'est une suite de calculs dont le résultat donne le *Plus Grand Commun Diviseur* de deux nombres.

Soit, plus généralement, deux nombres a et b. Tout nombre divisant à la fois a et b divise aussi leur différence. D'où l'idée que, pour trouver les diviseurs communs à deux nombres, on remplace leur recherche par celle des diviseurs communs à leur différence et au plus petit d'entre eux. Et on continue jusqu'à ce que l'on arrive à 0 ; le dernier nombre positif obtenu est le plus grand possible des diviseurs communs.

Voici ce que donne cet algorithme appliqué aux nombres 66 et 84 :

66 et 84	84 – 66 =18
66 et 18	66 – 18 = 48
	48 – 18 = 30
	30 – 18 = 12
18 et 12	18 – 12 = 6
12 et 6	12 – 6 = 6
6 et 6	6 – 6 = 0

Le PGCD de 66 et 84 est le nombre 6.

LE PROGRAMME DE L'ALGORITHME D'EUCLIDE

Voici le programme de calcul du PGCD de deux nombres A et B par l'algorithme d'Euclide, en MINI-langage (voir page suivante).
Lors de l'exécution de ce programme, l'ordinateur s'arrêtera pour que l'utilisateur puisse donner, par exemple, la valeur 66 à N, la valeur 84 à M et imprimera la phrase **6 est le PGCD de 66 et 84**.

18. 1 Le programme

```
entrer N, M
A = sup(N,M)
B = inf(N,M)
étiq1
R = A – B × int(A/B)
        si R <> 0
                A = B
                B = R
                allerà étiq1
        sinon
                sortir B, « est le PGCD de », N, « et », M
        finsi
```

Les instructions successives A = B, B = R, réalisent le remplacement de A et B par B et R.

Voyez ce que font les opérations *sup* et *inf*, ainsi que la fonction *int*, page suivante.

Note : plutôt que des soustractions successives d'un même nombre (comme le nombre 18, page précédente), il est plus rapide de faire la division entière (« euclidienne ») par ce nombre.

Quelques explications

Dans un programme, les instructions s'exécutent ligne après ligne, dans l'ordre de leur écriture. Les *indentations* des lignes permettent une meilleure lecture des programmes.

• L'instruction A = B signifie que la variable A prend la valeur de la variable B.

• L'instruction

allerà étiq4

commande d'exécuter l'instruction suivant l'*étiquette* étiq4 laissée quelque part dans le programme.

• L'instruction *conditionnelle* se comprend d'elle-même :

```
si condition
        [suite d'instructions effectuées si la condition est réalisée]
sinon
        [suite d'instructions effectuées sinon]
finsi.
```

si

sinon

finsi

L'écriture et l'exécution d'un programme

Un programme est d'abord écrit dans un *langage* particulier (comme notre *MINI-langage* ou *Python* ou *Scratch* ou *Java*) et mémorisé dans un ordinateur.
Si on a bien téléchargé ce langage, on peut alors l'exécuter par un ordre adéquat.
Pour éviter les erreurs, il est bon, avant de lancer l'exécution d'un programme, de *faire la machine* en exécutant, soi-même, sur le papier, les instructions successives.

18. 2 Un MINI-langage

Les programmes de ce chapitre sont écrits dans un MINI-langage, dont les 10 instructions n'utilisent que 14 mots français…

entrer, **sortir**
commentaire
allerà …, **étiqN**, **fin**
si *condition* **… sinon … finsi**
faire tantque *condition*, **finfaire**
procédure TRUC(A, B,… ; X, Y,…)
finprocédure

et les opérations et fonctions numériques élémentaires…

+, –, ×, / et ^ (pour la puissance)
sup(A, B) est le plus grand des nombres A ou B.
inf(A, B) est le plus petit des nombres A ou B.
int(X) est la partie entière du nombre X
rac(X) est la racine carrée de X
has(N) est un nombre choisi au hasard parmi 1, 2, … N
has(0) est un nombre à virgule compris entre 0 et 1.

Les conditions s'écrivent

A = B **A < B** **A > B** **A <= B** **A >= B**
A <> B (pour A différent de B).

procédure TRUC(A, B, … ; X, Y, …) est le début d'un programme, nommé TRUC, qui prend pour données les variables A et B (dont il interdit de changer les valeurs à l'intérieur de la procédure), et qui sort comme résultats les valeurs X et Y.
Il est conseillé de n'écrire que des procédures bien renseignées par des commentaires.

18. 3 Instructions de boucle

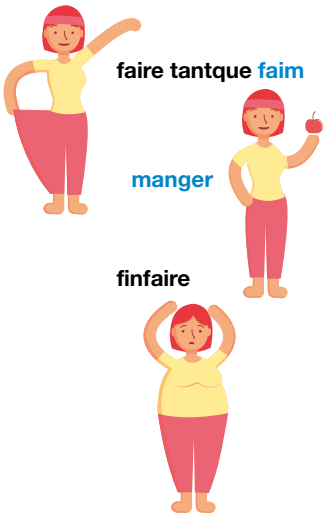
Une *boucle* réalise la répétition d'une certaine suite d'instructions (jusqu'à l'étiquette **finfaire**) tant qu'une certaine condition est réalisée (certaines variables changeant évidemment de valeur à chaque *itération*).
Si, à la fin de la boucle, cette condition n'est pas réalisée, on sort de la boucle en exécutant l'instruction située après la ligne finfaire.
Une boucle se présente ainsi :

faire tantque *condition*
*suite d'***instructions** *répétées tant que la condition reste réalisée*
finfaire

Dans la plupart des langages, il existe une autre instruction de boucle souvent utilisée :

faire pour X = A par pas de P jusqà R
*suite d'***instructions** *répétées tant que X est inférieur ou égal à R*
finfaire

Cette boucle s'exécute d'abord avec X = A, puis avec X = A + P,
puis avec X = A + 2P, … tant que X = A + k × P ⩽ R.
Cependant ce type de boucle est aussi réalisable avec l'instruction **fairetantque**, comme le montre le programme suivant.



Tous les diviseurs d'un nombre

```
procédure DIVISEURS(A)
        D = 1
        faire tantque D <= A/2
                Q = int(A/D)
                si Q × D = A
                        sortir D
                sinon
                finsi
        D = D + 1
        finfaire
finprocédure
```

Avec A = **1001**,
la procédure DIVISEURS sort :

1
7
11
13
77
91
143

En arithmétique (et ailleurs), il suffit souvent de modifier légèrement un programme pour lui faire faire bien autre chose.
Voyez, ci-dessous, comment modifier le programme précédent pour décomposer un nombre en facteurs premiers et comment sortir une liste de nombres premiers.
À vous de faire les petits programmes suivants :
• Deux nombres étant donnés, dire si l'un est divisible par l'autre.
• Un nombre étant donné, dire s'il est premier.

Facteurs premiers d'un nombre

```
procédure FACTEURSPREMIERS(A)
M = rac(A)
B = A
D = 2
faire tantque D <= M
        Q = int(B/D)
        si Q × D = B
                sortir D
                B = Q
        sinon
        D = D + 1
        finsi
finfaire
finprocédure
```

Avec A = **60**,
la procédure FACTEURSPREMIERS sort :

2
2
3
5

Liste des nombres premiers

```
procédure PREMIERS(N)
D = 2
        faire pour A = 2 par pas de 1 jusquà N
                faire tantque D <= rac(A)
                        Q = int(A/D)
                                si Q × D = A
                                A = A + 1
                                finsi
                        D = D + 1
                finfaire
        sortir A
        finfaire
finprocédure
```

Avec N = **100**,
La procédure PREMIERS sort :

2	**3**	**5**	**7**
11	**13**	**17**	**19**
23	**29**	**31**	**37**
41	**43**	**47**	**53**
59	**61**	**67**	**71**
73	**79**	**83**	**89**
97			

Le jeu du piquet à cheval tire son nom d'un jeu de cartes, le piquet, et d'une mise en scène où deux cavaliers, voulant jouer tout en voyageant, jouent donc sans cartes. Nous avons choisi la variante du jeu où il faut atteindre 32 en ajoutant entre 1 et 4 au nombre précédent.

Règle du jeu :
Le premier joueur dit un nombre entre 1 et 4.
Puis chaque joueur à son tour dit un nombre égal au précédent augmenté de 1 à 4 unités.
Celui qui dit 32 a gagné.

Programmer un jeu : Le jeu du piquet à cheval

Voici le programme du jeu, dans lequel la machine gagne toujours contre l'utilisateur.

```
procédure COURSE
    N = 32
    P = 4
    M = 2
    sortir « je joue le premier et je dis », M
    étiq1
    sortir « que dites-vous ? »
    entrer V
        si V > M + 4
        sortir « réponse interdite ! »
        allerà étiq1
        sinon
        M = M + P + 1
        sortir « vous aviez dit : », V, « je dis : », M
        finsi
            si M = N
            sortir « j'ai gagné »
            sinon
            allerà étiq1
            finsi
finprocédure
```

Plus généralement, on peut jouer au
JEU de la COURSE à N par PAS MAXIMUM de P.

À vous de modifier le programme précédent de manière à ce que la machine gagne toujours. Si N n'est pas multiple de P+1, elle doit alors dire d'abord le nombre M égal à $N - \text{int}(N/(P+1)) \times (P+1)$, puis ajouter P + 1 au nombre qu'elle avait dit juste avant. Et si N est multiple de P+1, il faut faire commencer le joueur puis la machine dit P+1 et poursuivre comme précédemment.
Ensuite vous pourrez vous attaquer à la programmation du jeu laissant le choix de commencer ou non à l'utilisateur. Il vous faudra alors, dans le cas où celui-ci connaîtrait la stratégie gagnante, faire jouer la machine au hasard, jusqu'à ce que l'utilisateur se trompe, ou bien jusqu'à faire accepter la défaite de la machine.
Si vous disposez alors de dessins ou de sons, n'hésitez pas à faire exprimer ses « sentiments » par la machine.

Un jeu de piquet,
Ernest Meissonier

Les ordinateurs sont aujourd'hui très utilisés pour faire des simulations de phénomènes réels, comme la trajectoire d'une fusée, la diffusion d'un médicament dans l'organisme, le revenu d'une taxe, une partie de jeu d'échecs ou l'évolution du climat.
Ces simulations permettent, en effet, de remplacer des expériences, souvent coûteuses (en temps ou en argent), et d'étudier les effets de la variation de différents paramètres conduisant le phénomène.

Pour simuler un phénomène, il faut disposer d'un **modèle mathématique** *qui en traduit les caractéristiques. Cela oblige à préciser les paramètres qui le définissent et les hypothèses qui le rendent suffisamment fidèle à la réalité.*

Une simulation d'épidémie

Voici par exemple une simulation d'épidémie par un modèle simplifié (et sans possibilité de vaccination) mais dont on peut tirer d'efficaces enseignements. En voici les données et hypothèses.

• Le phénomène est étudié en des dates mesurées par des nombres entiers :
0, 1, 2, 3, …

• Le temps d'étude est suffisamment court pour que la population totale puisse être considérée comme constante (il n'y a pas de naissances).

• La population appartient à 4 catégories :
x : les individus non atteints et non immunisés,
y : les malades,
z : les individus immunisés (naturellement ou après une heureuse issue de la maladie),
t : les morts.

On choisit $x + y + z + t = 100$.

D'une étape à l'autre…

… le nombre de nouveaux malades est proportionnel, à la fois, au nombre de malades et au nombre de personnes susceptibles d'être contaminés :

$$x_{n+1} = x_n - a \times x_n \times y_n$$

$a = 1/100$ en vertu de l'argument de la contagion limite : si on adjoint un non malade, non immunisé, à une population entièrement composée de malades, il est certain de tomber malade.

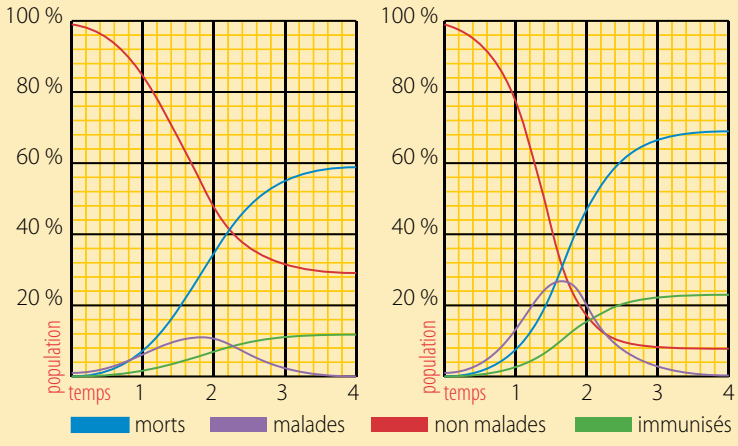
… le taux de malades qui guérissent (et sont alors immunisés) est b ($0 < b < 1$) et le taux de malades qui meurent est c ($0 < c < 1$) :

$$z_{n+1} = z_n + b \times y_n$$
$$t_{n+1} = t_n + c \times y_n$$
$$y_{n+1} = (1 - b - c) \times y_n + a \times x_n \times y_n$$

Notez que les termes des suites comme x_n sont ici notés X(N).

procédure EPIDEMIE(B, C ; X, Y, Z, T)
X(0) = 99
Y(0) = 1
Z(0) = 0
T(0) = 0
faire pour N = 0 **par pas de** 1 **jusqu'à** Q – 1
X(N + 1) = X(N) – X(N) × Y(N)/100
Z(N + 1) = Z(N) + B × Y(N)
T(N + 1) = T(N) + C × Y(N)
Y(N + 1) = (1 – B – C) × Y(N) + X(N) × Y(N)/100
finfaire
finprocédure

La procédure EPIDEMIE calcule les valeurs de x_n, y_n, z_n et t_n pour n valant de 1 à Q.



Si l'on dispose d'un programme traçant les courbes donnant les valeurs d'une suite, on obtient le genre de résultats intéressants ici montrés (on a pris Q = 20)…
Les premières courbes sont obtenues pour $b = 0{,}1$ et $c = 0{,}5$;
les deuxièmes pour $b = 0{,}1$ et $c = 0{,}3$.
On y voit que, avec ce modèle (sans mesures d'isolement ni possibilité de soins), la diminution du taux de mortalité fait finalement augmenter le nombre de malades… et de morts !