Petit cours d'algorithmique

Michel Ramus

On peut définir la notion d'**algorithme** comme une suite d'instructions à exécuter pour obtenir un résultat voulu.

A - Algorithmes et vie quotidienne

Vous avez sans doute déjà fait de l'algorithmique sans le savoir.

En effet, les recettes de cuisine et les notices de montage fournissent des exemples simples d'algorithmes qui vous ont permis d'obtenir des résultats plus ou moins réussis.

Lorsqu'une personne vous demande de lui indiquer le chemin à suivre pour aller à un endroit donné, vous pouvez lui dessiner un plan, mais il faut que la personne soit capable de le lire.
Pour beaucoup, il sera plus facile de suivre des instructions telles que : dirigez-vous vers … , prenez la deuxième à gauche, …

Dans un jeu de société, si vous devez communiquer à une personne des informations lui permettant de construire une figure identique à une figure qu'elle ne voit pas, dans la plupart des cas, une liste d'instructions à exécuter sera plus efficace qu'une description de la figure.

B - Algorithmes et mathématiques

Voici un algorithme qui permet de ranger des nombres écrits sur une feuille dans l'ordre décroissant :
- Chercher et barrer le plus grand nombre.
- Ecrire ce nombre.
- Recommencer avec les nombres non barrés.

Quelle que soit la quantité de nombres à ranger, trois d'instructions suffisent, car la troisième relance le processus jusqu'à ce qu'il n'y ait plus de nombres à ranger.
Ceci fournit un exemple d'algorithme **récursif**, c'est à dire, d'algorithme qui fait appel à lui même.

Voici un algorithme **récursif** qui permet d'écrire les nombres entiers inférieurs à 4000 en chiffres romains.
Pour écrire un nombre :
- Chercher le plus grand nombre de l'échelle contenu dans ce nombre.
- Ecrire ce plus grand nombre en chiffres romains.
- Soustraire ce plus grand nombre du nombre à écrire et recommencer avec le reste.

Echelle de nombres à utiliser avec l'algorithme

M	CM	D	CD	C	XC	L	XL	X	IX	V	IV	I
1000	900	500	400	100	90	50	40	10	9	5	4	1

Pour 1947, cela donne successivement :
M reste 947
MCM reste 47
MCMXL reste 7
MCMXLV reste 2
MCMXLVI reste 1
MCMXLVII

L'algorithme suivant, qui lui aussi est **récursif**, permet de déchiffrer les inscriptions en chiffres romains.
Pour déchiffrer une inscription en chiffres romains :
- Si la valeur du premier chiffre de gauche est inférieure à celle du chiffre suivant, écrire la différence des deux valeurs et barrer les deux chiffres, sinon écrire la valeur du premier chiffre et le barrer.
- Recommencer tant qu'il y a des chiffres non barrés, puis additionner toutes les valeurs écrites.
Pour MCMXLV cela donne successivement :
- Ecrire 1000, barrer M
- Ecrire 1000 - 100 = 900, barrer CM
- Ecrire 50 - 10 = 40, barrer XL
- Ecrire 5, barrer V
- 1000 + 900 + 40 + 5 = 1945

Ces exemples montrent que l'élaboration d'algorithmes permet d'automatiser des tâches complexes en les réduisant à l'exécution de suites d'instructions élémentaires.
Une approche algorithmique fournit ainsi un moyen d'augmenter les chances de réussite, c'est aussi un premier pas vers la **programmation**.

C - Algorithmes et programmation

Programmer, c'est trouver des algorithmes exécutables par un ordinateur et les traduire dans un langage de programmation.

Programmation du calcul des puissances entières d'un nombre non nul

Voici la définition classique de la puissance nième d'un nombre.
a étant un nombre non nul et n un entier naturel
si n = 0 alors $a^n = 1$
si n = 1 alors $a^n = a$
si n > 1 alors a^n est le produit de n facteur égaux à a
Cela donne
$5^3 = 5 \times 5 \times 5$, mais pour continuer il faut savoir calculer un produit de plus de deux nombres.

En voici une définition **récursive**.
a étant un nombre non nul et n un entier naturel
si n = 0 alors $a^n = 1$
si n = 1 alors $a^n = a$
si n > 1 alors $a^n = a \times a^{n-1}$
Pour 5^3, cela donne successivement
$5^3 = 5 \times 5^2$
$= 5 \times (5 \times 5^1)$
$= 5 \times (5 \times 5)$
$= 5 \times 25$
$= 125$
L'application de cette définition déclenche un processus automatique qui aboutira si l'on sait multiplier deux nombres.
Les ordinateurs sont capables de mener à bien ce type de processus.

Voici comment on peut traduire cette définition dans le langage de programmation **Revolution**[1]

```
function aexposantn a n
  if n = 0 then return 1
  if n = 1 then return a
  else return a * aexposantn (a, n-1)
end aexposantn
```

Les lignes ci-dessus définissent une fonction nommée aexposantn qui a deux variables a et n, et qui retourne la valeur de a exposant n.
En effet, en appelant cette fonction avec les valeurs 5 et 3, on déclenche le processus suivant :
aexposantn (5, 3) retourne 5 x aexposantn (5, 2)
retourne 5 x (5 x aexposantn (5, 1))
retourne 5 x (5 x 5)
retourne 5 x 25
retourne 125

aexposantn est le nom donné à la fonction par le programmeur.
function, if, then, else, return, end sont des mots du langage **Revolution**

Et voici la définition de la même fonction dans le langage de programmation **xlogo**[2]

```
pour aexposantn :a :n
si :n = 0 [retourne 1]
si :n = 1 [retourne :a]
retourne :a * aexposantn :a :n-1
fin
```

aexposantn est le nom donné à la fonction par le programmeur.
Pour, si, retourne, fin sont des mots du langage **xlogo**

Pour obtenir la valeur de 5^3, il suffit de taper aexposantn 5 3
En effet :
aexposantn 5 3 retourne 5 x aexposantn 5 2
retourne 5 x (5 x aexposantn 5 1)
retourne 5 x (5 x 5)
retourne 5 x 25
retourne 125

Programmation de l'écriture d'un nombre en chiffres romains

Voici une traduction dans le langage **Revolution** de l'algorithme donné pour écrire un nombre entier inférieur à 4000 en chiffres romains :

[1] Langage multiplateforme téléchargeable gratuitement (lancer une recherche runrev)
[2] Langage multiplateforme téléchargeable gratuitement (lancer une recherche xlogo)

```
function rom n tra
  if n = 0 then return tra
  if n > 999 then return rom(n - 1000, tra & "M")
  if n > 899 then return rom(n - 900, tra & "CM")
  if n > 499 then return rom(n - 500, tra & "D")
  if n > 399 then return rom(n - 400, tra & "CD")
  if n > 99 then return rom(n - 100, tra & "C")
  if n > 89 then return rom(n - 90, tra & "XC")
  if n > 49 then return rom(n - 50, tra & "L")
  if n > 39 then return rom(n - 40, tra & "XL")
  if n > 9 then return rom(n - 10, tra & "X")
  if n > 8 then return rom(n - 9, tra & "IX")
  if n > 4 then return rom(n - 5, tra & "V")
  if n > 3 then return rom(n - 4, tra & "IV")
  if n > 0 then return rom(n - 1, tra & "I")
end rom
```

Nous définissons ainsi une fonction nommée rom à deux variables n et tra qui sont respectivement le nombre à écrire et la chaîne de caractères à construire pour obtenir l'écriture en chiffres romains.
L'opérateur & retourne la chaîne de caractères obtenue en **concaténant** les deux chaînes qui l'entourent.
"M" & "CM" retourne "MCM"
La fonction rom appelée avec le nombre à écrire et une chaîne vide retourne la chaîne construite qui est l'écriture du nombre en chiffres romains.
En effet :

```
rom ( 46, "")   retourne rom (6, "XL")
                retourne (retourne rom (1, "XLV"))
                retourne (retourne (retourne rom (0, "XLVI")))
                retourne (retourne (retourne "XLVI")
                retourne (retourne "XLVI")
                retourne "XLVI"
```

Voici la définition de la même fonction dans le langage **xlogo** :

```
pour rom :n :tra
si :n=0 [retourne :tra]
si :n>999 [retourne rom :n-1000 mot :tra "M]
si :n>899 [retourne rom :n-900 mot :tra "CM]
si :n>499 [retourne rom :n-500 mot :tra "D]
si :n>399 [retourne rom :n-400 mot :tra "CD]
si :n>99 [retourne rom :n-100 mot :tra "C]
si :n>89 [retourne rom :n-90 mot :tra "XC]
si :n>49 [retourne rom :n-50 mot :tra "L]
si :n>39 [retourne rom :n-40 mot :tra "XL]
si :n>9 [retourne rom :n-10 mot :tra "X]
si :n>8 [retourne rom :n-9 mot :tra "IX]
si :n>4 [retourne rom :n-5 mot :tra "V]
si :n>3 [retourne rom :n-4 mot :tra "IV]
si :n>0 [retourne rom :n-1 mot :tra "I]
fin
```

Dans ce langage c'est l'opérateur mot qui assure la **concaténation** de deux chaînes de caractères.
mot "M "CM retourne "MCM
Pour obtenir l'écriture de 46 il suffit de taper rom 46 "

En effet :

```
rom 46 "    retourne rom 6 "XL
            retourne (retourne rom 1 "XLV)
            retourne (retourne (retourne rom 0 "XLVI))
            retourne (retourne (retourne "XLVI))
            retourne (retourne "XLVI)
            retourne "XLVI
```

Ces exemples montrent que le même algorithme peut se traduire dans différents langages et que les différences entre les programmes se situent à d'autres niveaux.
Par exemple :

- Les mêmes instructions peuvent se retrouver avec des noms différents
  if et si, return et retourne, end et fin.
- Le même objet peut se noter de façons différentes
  "CM" et "CM pour la chaîne de caractères CM
  tra et :tra pour la variable tra
- Le même opérateur peut se retrouver avec une désignation et une syntaxe différentes
  "M" & "CM" et mot "M "CM pour la concaténation des chaînes de caractères tra et CM
- Le mode de définition d'une fonction et la façon de l'appeler peuvent être différents
  aexposantn (5,3) et aexposantn 5 3 pour le calcul de 5^3

D - Conclusion

L'algorithmique permet de simplifier et d'automatiser certaines tâches de la vie courante et du domaine mathématique en les réduisant à l'exécution de suites d'instructions élémentaires.
C'est aussi l'algorithmique qui permet de programmer les ordinateurs. En effet, les langages de programmation ne sont que les moyens qui permettent de traduire les algorithmes mis au point pour obtenir les résultats voulus. L'apprentissage de la programmation ne peut donc pas se réduire à celui d'un ou de plusieurs langages.
Les quelques exemples donnés mettent en évidence le caractère récursif de certains algorithmes et son intérêt.
Ils laissent entrevoir la concision et la puissance des langages de programmation.