L'essentiel à connaître en algorithmique

1 La notion d'état de la mémoire et de variable

On peut se représenter la mémoire d'un ordinateur comme étant formée d'une multitude de petites boîtes qui ont chacune un nom et qui contiennent chacune une donnée (par exemple un nombre). Une boîte à laquelle on a donné un nom est appelé une **variable**.

Attention : cours de l'exécution d'un programme, le contenu des boîtes peut être modifié, on dira que l'on modifie le contenu de la variable.

- A← 1
- A= 1 *signifient que **la variable A** prend la valeur 1*
- A prend la valeur 1

Remarque très importante : Certains langages ou programmes demandent à ce que les variables soient déclarées au début (c'est le cas pour Algobox), d'autres non, comme par exemple pour la programmation de votre calculatrice.

2. La structure globale de l'algorithme ou du programme

Un algorithme est écrit en langage naturel ce qui lui permet d'être lisible de façon assez aisée par un être humain. Il suit de façon précise la structure **du programme**, qui sera écrit dans un langage de programme, afin d'être éxecuté par une machine numérique. La différence essentielle entre l'algorithme et le programme réside donc dans le destinataire (**humain ou machine**) et non dans le degré de complexité.

Les éléments principaux sont les suivants :

1. La déclaration de variables, et le typage des données

Ici on définit le nom des variables et surtout leur type (nombre entier, nombre décimal, tableau, chaine de caractères, booléen…). Des noms de variables peuvent être interdits, par exemple, il est impossible de donner à une variable le nom d'une instruction, d'une fonction prédéfinie. Le respect de certaines règles permet une meilleure lisibilité des codes.

2. Les Input

Ici on définit ce que le programme reçoit de l'extérieur (données externes, actions de l'utilisateur…)

3. Les Output

C'est ce que le programme doit renvoyer à l'utilisateur (affichage, action…)

4. Le cœur de l'algorithme

Il permet le traitement de l'information et est composé d'instructions prédéfinies.

Dans certains langages de programmation, il est nécessaire d'effectuer une opération de compilation du code car la machine « ne comprend » pas directement le langage de programmation (par exemple C, C++). D'autres langages sont dits « interprétables », ils ne nécessitent pas de compilation, ce sont des logiciels qui traitent en leur propre sein le code (par exemple Python).

3. Les instructions

a. Les instruction d'Entrée Sortie (Input/Output)

LIRE… Lit la valeur d'une variable ECRIRE… Écrit la valeur d'une valeur ou d'un texte

b. La séquence

Définition : Une séquence est une instruction qui sert à enchaîner deux instructions .

Une séquence écrit :

```
Instruction1 ;
Instruction2 ;
```

Remarques :

Dans les langages de programmation on utilise souvent des séparateurs : « ; » et « {} » afin que le programme puisse reconnaître la structure. Dans certains langages de programmation, la position des instructions sur la ligne et entre les lignes permet de les hiérarchiser (Python). On parle dans ce cas de langage de programmation à indentation.

A l'aide de l'instruction séquence on peut en fait enchaîner une infinité d'instructions.

c. L'affectation

Définition : Une **affectation** est une instruction qui sert à modifier le contenu de l'une des variables de la mémoire.

Une affectation s'écrit :

- **x=expression ou nombre** ou x← expression ou nombre ou x prend la valeur expression ou nombre;
- **x** est le nom d'une variable.
- **expression** est une expression algébrique formée avec les noms des variables et les quatre opérations.

Remarque la plupart des logiciels donnent accès à des fonctions toutes faites comme la racine carrée de la fonction puissance.

```
x=1 ;
y=2 ;
z=2*x+3*y ;
x=z ;
```

Dans cette suite d'affectations, la valeur des variables est la suivante :

x=1
y=2
z=7
x=7

Attention : le contenu de la variable z ne sera modifié que si l'on demande un recalcul. C'est-à-dire que si l'on fait afficher le contenu de **z** à la suite de ces instructions, la modification du contenu de la variable **x** ne sera pas automatiquement répercutée sur la variable **z**.

```
1 VARIABLES

2 x EST_DU_TYPE NOMBRE
3 y EST_DU_TYPE NOMBRE
4 z EST_DU_TYPE NOMBRE

5 DEBUT_ALGORITHME

6 x PREND_LA_VALEUR 1
7 y PREND_LA_VALEUR 2
8 z PREND_LA_VALEUR 2*x+3*y
9 AFFICHER "Avant modification de x : z= "
10 AFFICHER z
11 x PREND_LA_VALEUR z
12 AFFICHER "x="
13 AFFICHER x
14 AFFICHER "Après modification de x : z= "
15 AFFICHER z

16 FIN_ALGORITHME
```

Traitement de l'algorithme :

```
***Algorithme lancé***
Avant modification de x :  z= 8
x=8
Après modification de x :  z= 8
***Algorithme terminé***
```

d. L' instruction conditionnelle ou le test

Définition : un test est une instruction qui sert à exécuter une instruction si une condition est vraie et éventuellement une autre instruction si la condition est fausse.

Un test s'écrit :

```
SI…
SINON….FIN
SI
```

Le SI est suivi d'une proposition vraie ou fausse appelée TEST. Par exemple $n > 0$ ou $a - b > 10$. Si le test est vrai alors l'instruction suivante est exécutée sinon, c'est celle suivant le SINON qui l'. Il est possible d'omettre le SINON dans le cas où il n'y a aucune opération à effectuer dans ce deuxième cas.

```
SI TEST INSTRUCTION A
SINON INSTRUCTION B
FIN SI
```

Si TEST est vrai alors l'instruction A est effectué et s'il est faux alors c'est l'instruction B qui le sera.

Exemple : La valeur absolue : $|X| \begin{cases} Si\ X < 0, écrire - X \\ Sinon, écrire\ X \end{cases} Fin\ Si$

e. Les boucles

Définition : Une boucle est une instruction qui sert à répéter une instruction plusieurs fois.

Il existe deux types de boucle :

- La boucle TantQue, que l'on utilise lorsque l'on ne connaît pas a priori le nombre d'exécutions de la boucle ;
- La boucle Pour, version condensée de la boucle TantQue, que l'on utilise lorsqu'on connaît à l'avance le nombre d'exécutions de la boucle.

- **La boucle Tantque**

TANT QUE… FAIRE… FIN TANT

Tant que le test est vrai, l'instruction qui suit est effectuée.

Il s'agit d'une **boucle conditionnelle**.

Exemple :

```
x=1 ;
TantQue x<100 faire
        x=x*2
Fin TantQue
```

- **La boucle itérative ou boucle Pour**

```
POUR Nom_de_Variable ALLANT DE Valeur_Initiale JUSQU'A Valeur_Finale
FAIRE…
FIN POUR
```

On répète un nombre fixé de fois, une instruction. On incrémente de 1 une variable associée à un compteur. La difficulté principale provient de la valeur initiale que l'on donne à ce compteur et de la valeur d'arrêt. La valeur d'initialisation des variables intervenant après le Faire est primordiale, tout comme l'ordre des instructions.

Exemple :

```
x=1 ;
Pour i allant de 1 à 3 faire

        x=x*2

FinPour

Afficher x ;
```

Comparez l'algorithme précédent avec les suivants :

```
x=1 ;
i=1 ;
TantQue i≤3 faire

        x=x*2
        i=i+1
Fin TantQue

Afficher x;
```

```
x=1 ;
i=0 ;
TantQue i<3 faire

        x=x*2
        i=i+1
Fin TantQue

Afficher x;
```

f. Un type particulier de données: le tableau

Définition : Un tableau est une boîte contenant plusieurs cases numérotées. Elles peuvent contenir des nombres, des caractères ou des chaînes de cracatères.

Attention : Selon les logiciels utilisés (surtout les calculatrices) la numérotation ne commence pas à zéro mais à un.

```
Pour i allant de 0 à n-1 faire

        t[i]=i^2

Finpour
```

L'algorithme précédent affecte les valeurs suivantes :

t[0]=0
t[1]=1
t[2]=4
…..
t[n-1]=(n-1)²

```
Pour i allant de 1 à n faire

        t[i]=i^2

Finpour
```

L'algorithme précédent affecte les valeurs suivantes :

t[1]=1
t[2]=4
t[3]=9
…..
t[n]=n²