NOTIONS D'ALGORITHMIQUE

I] Généralités

Le **langage informatique** est celui de l'ordinateur auquel l'homme confie un travail. Il en existe différents avec lesquels on écrit des programmes: Basic, Pascal, Fortran, Python, Java, C, C++, Cobol…

Le **langage algorithmique** est utilisé en amont, avant le codage dans un langage pour décrire la succession d'opérations à effectuer par le programme.

Définition: Un algorithme est une succession d'opérations logiques ou de calculs présentés dans l'ordre chronologique de leur exécution.

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 exécuté 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.

Structure globale de l'algorithme ou du programme

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. L'entrée

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

3. Le cœur de l'algorithme : la suite d'instructions

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

4. Les sorties

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

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).

Signalons aussi:

L'**INITIALISATION** des variables qui permet de leur donner une valeur initiale. Dans certains langages, elle est impérative.

L'**INCRÉMENTATION** est un terme définissant le "saut" que font les variables numériques augmentées d'un pas h. Par exemple on incrémente la variable n d'une unité.

II] Les instructions usuelles

- **Les instructions d'entrée /sortie (Input/Output)**

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

- **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.

- **Affectation**

Elle consiste à donner une valeur à une variable.

$n \leftarrow a$ signifie que la variable n prend la valeur a. On écrit souvent dans les programmes : $n = a$ qui est différent de $a = n$ contrairement aux mathématiques. Dans ce dernier cas, c'est la variable a qui prend la valeur n.

On écrirait dans l'algorithme $a \leftarrow n$.

Exemples :

$n \leftarrow 1$

$n \leftarrow n + 1$

$j \leftarrow i + 1$

La deuxième instruction correspond à l'incrémentation de la variable n d'un pas de 1.

- **Les instructions conditionnelles**

```
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. Il est possible d'omettre le SINON dans le cas où il n'y a aucune opération à effectuer dans ce deuxième cas.

Exemple :

```
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.

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

$Fin\ Si$

```
TANT QUE…
FAIRE…
FIN TANT QUE…
```

Tant que le test est vrai, l'instruction qui suit est effectuée. Il s'agit d'une **boucle conditionnelle**.

- **La boucle itérative**

```
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.

On peut remarquer cela dans les deux algorithmes suivants du calcul de la somme des premiers entiers naturels.

L'Algorithme Algobox renvoit pour n=5:

```
1   VARIABLES
2     i EST_DU_TYPE
NOMBRE
3     s EST_DU_TYPE
NOMBRE
4     n EST_DU_TYPE
NOMBRE
5   DEBUT_ALGORITHME
6     AFFICHER "Calcul de
1+2+3+...n"
7     LIRE n
8     s PREND_LA_VALEUR 0
9     POUR i ALLANT_DE 1
A n
10      DEBUT_POUR
11      s PREND_LA_VALEUR
s+i
12      AFFICHER s
13      FIN_POUR
14  FIN_ALGORITHME
```

```
***Algorithme lancé***
Calcul de 1+2+3+...n
1
3
6
10
15
***Algorithme terminé***
```

En inversant les lignes 11 et 12, on n'obtient pas l'affichage de la somme pour la dernière valeur de n. De plus la première valeur de la somme affichée est 0 car l'instruction d'affichage précède le calcul de la somme.

```
1   VARIABLES
2     i EST_DU_TYPE
NOMBRE
3     s EST_DU_TYPE
NOMBRE
4     n EST_DU_TYPE
NOMBRE
5   DEBUT_ALGORITHME
6     AFFICHER "Calcul de
1+2+3+...n"
7     LIRE n
8     s PREND_LA_VALEUR 0
9     POUR i ALLANT_DE 1
A n
10      DEBUT_POUR
11      AFFICHER s
12      s PREND_LA_VALEUR
s+i
13      FIN_POUR
14  FIN_ALGORITHME
```

```
***Algorithme lancé***
Calcul de 1+2+3+...n
0
1
3
6
10
***Algorithme terminé***
```