ALGORITHMIQUE

« A l'origine de toute erreur attribuée à l'ordinateur, vous trouverez au moins deux erreurs humaines. Dont celle consistant à attribuer l'erreur à l'ordinateur. » - Anonyme

Introduction :

Un algorithme, c'est une SUITE D'INSTRUCTIONS, qui une fois exécutée correctement, conduit à un résultat donné.

Si l'algorithme est juste, le résultat est le résultat voulu.

Pour fonctionner, **un algorithme doit contenir uniquement des INSTRUCTIONS COMPREHENSIBLES PAR CELUI QUI DEVRA L'EXECUTER**.

Les ordinateurs, quels qu'ils soient, ne sont fondamentalement capables de comprendre que **QUATRE CATEGORIES D'INSTRUCTIONS). Ces quatre familles d'instructions sont :**

- **L'AFFECTATION DE VARIABLES**
- **LA LECTURE / ECRITURE**
- **LES TESTS**
- **LES BOUCLES**

Un algorithme informatique se ramène donc toujours au bout du compte à la combinaison de ces quatre briques de base.

ALGORITHMIQUE - Partie 1
Les Variables

A quoi servent les variables ?

Dans un programme informatique, on va avoir en permanence besoin de stocker provisoirement des valeurs. Il peut s'agir de données issues du disque dur, fournies par l'utilisateur (frappées au clavier), ou que sais-je encore. Il peut aussi s'agir de résultats obtenus par le programme, intermédiaires ou définitifs. Ces données peuvent être de plusieurs types (on en reparlera) : elles peuvent être des nombres, du texte, etc. Toujours est-il que dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une **variable**.

Une variable est une **boîte**, que le programme va repérer par une **étiquette**. Pour avoir accès au contenu de la boîte, il suffit de la désigner par son étiquette.

Dans l'ordinateur, physiquement, il y a un emplacement de mémoire, repéré par une adresse binaire. Si on programmait dans un langage directement compréhensible par la machine, on devrait se fader de désigner nos données par de superbes 10011001 et autres 01001001 (enchanté !). Mauvaise nouvelle : de tels langages existent ! Ils portent le doux nom d'assembleur. Bonne nouvelle : ce ne sont pas les seuls langages disponibles. Autre « bonne » nouvelle : On travaillera aussi cette année en assembleur 😊

Les langages informatiques plus évolués (ce sont ceux que presque tout le monde emploie) se chargent précisément, entre autres rôles, d'épargner au programmeur la gestion fastidieuse des emplacements mémoire et de leurs adresses. Et, comme vous commencez à le comprendre, il est beaucoup plus facile d'employer les étiquettes de son choix, que de devoir manier des adresses binaires.

Déclaration des variables

La première chose à faire avant de pouvoir utiliser une variable est de **créer la boîte et de lui coller une étiquette**. Ceci se fait tout au début de l'algorithme, avant même les instructions proprement dites. C'est ce qu'on appelle la **déclaration des variables**.

Le **nom** de la variable (l'étiquette de la boîte) obéit à des impératifs changeant selon les langages. Toutefois, une règle absolue est qu'un nom de variable peut comporter des lettres et des chiffres, mais qu'il exclut la plupart des signes de ponctuation, en particulier les espaces. Un nom de variable correct commence également impérativement par une lettre.

Lorsqu'on déclare une variable, il ne suffit pas de créer une boîte (réserver un emplacement mémoire) ; encore doit-on préciser ce que l'on voudra mettre dedans, car de cela dépendent la **taille** de la boîte (de l'emplacement mémoire) et le **type de codage** utilisé.

Types numériques classiques

Commençons par le cas très fréquent, celui d'une variable destinée à recevoir des nombres.

Si l'on réserve un octet pour coder un nombre, on ne pourra coder que $2^8 = 256$ valeurs différentes. Cela peut signifier par exemple les nombres entiers de 1 à 256, ou de 0 à 255, ou de –127 à +128… Si l'on réserve deux octets, on a droit à 65 536 valeurs ; avec trois octets, 16 777 216, etc. Et là se pose un autre problème : ce codage doit-il représenter des nombres décimaux ? des nombres négatifs ?

Bref, le type de codage (autrement dit, le type de variable) choisi pour un nombre va déterminer :

- les valeurs maximales et minimales des nombres pouvant être stockés dans la variable
- la précision de ces nombres (dans le cas de nombres décimaux).

Tous les langages, quels qu'ils soient offrent un « bouquet » de types numériques, dont le détail est susceptible de varier légèrement d'un langage à l'autre. Généralement, on retrouve cependant les types suivants :

Type Numérique	Plage
Byte (octet)	0 à 255
Entier simple	-32 768 à 32 767
Entier long	-2 147 483 648 à 2 147 483 647
Réel simple	$-3{,}40\times10^{38}$ à $-1{,}40\times10^{45}$ pour les valeurs négatives $1{,}40\times10^{-45}$ à $3{,}40\times10^{38}$ pour les valeurs positives
Réel double	$1{,}79\times10^{308}$ à $-4{,}94\times10^{-324}$ pour les valeurs négatives $4{,}94\times10^{-324}$ à $1{,}79\times10^{308}$ pour les valeurs positives

Pourquoi ne pas déclarer toutes les variables numériques en réel double, histoire de bétonner et d'être certain qu'il n'y aura pas de problème ? En vertu du principe de ***l'économie de moyens****. Un bon algorithme ne se contente pas de « marcher » ; il marche en évitant de gaspiller les ressources de la machine. Sur certains programmes de grande taille, l'abus de variables surdimensionnées peut entraîner des ralentissements notables à l'exécution, voire un plantage pur et simple de l'ordinateur. Alors, autant prendre dès le début de bonnes habitudes.*

En algorithmique, on ne se tracassera pas trop avec les types. On se contentera parfois de préciser qu'il s'agit d'un nombre, en gardant en tête que dans un vrai langage, il faudra être plus précis.

En pseudo-code, une déclaration de variables se fera ainsi :

```
Variable g en Numérique
ou encore
Variables PrixHT, TauxTVA, PrixTTC
```

Type alphanumérique

Les boîtes que sont les variables peuvent contenir bien d'autres informations que des nombres. On dispose donc également du **type alphanumérique** (également appelé **type caractère**, **type chaîne** ou en anglais, le **type char et type string**

Dans une variable de ce type, on stocke des **caractères**, qu'il s'agisse de lettres, de signes de ponctuation, d'espaces, ou même de chiffres. Le nombre maximal de caractères pouvant être stockés dans une seule variable **string** dépend du langage utilisé.

En pseudo-code, une chaîne de caractères est toujours notée entre guillemets

L'instruction d'affectation

Syntaxe et signification

On ne peut pas faire trente-six mille choses avec une variable, mais seulement une et une seule.
Cette seule chose qu'on puisse faire avec une variable, c'est **l'affecter,** c'est-à-dire **lui attribuer une valeur**. On peut « remplir la boîte ».

En pseudo-code, l'instruction d'affectation se note avec le signe ←

Ainsi :

```
Bidule ← 24
```

Attribue la valeur 24 à la variable Bidule.
Ceci sous-entend impérativement que Bidule soit une variable de type numérique. Si Bidule a été défini dans un autre type, il faut bien comprendre que cette instruction provoquera une erreur
On peut aussi attribuer à une variable la valeur d'une autre variable, telle quelle ou modifiée. Par exemple :

```
Truc ← Bidule
```

Signifie que la valeur de Truc est maintenant celle de Bidule.
Notez bien que cette instruction n'a en rien modifié la valeur de Bidule : **une instruction d'affectation ne modifie que ce qui est situé à gauche de la flèche**. Notez aussi que si après on modifie la valeur de Bidule, cela ne modifiera pas Truc

```
Tutu ← Toto + 4
```

Si Toto contenait 12, Tutu vaut maintenant 16. De même que précédemment, Toto vaut toujours 12.

```
Tutu ← Tutu + 1
```

Si Tutu valait 6, il vaut maintenant 7. La valeur de Tutu est modifiée, puisque Tutu est la variable située à gauche de la flèche.
Pour revenir à présent sur le rôle des guillemets dans les chaînes de caractères et sur la confusion numéro 2 signalée plus haut, comparons maintenant deux algorithmes suivants :

```
Exemple n°1
Début
Riri ← "Loulou"
Fifi ← "Riri"
Fin


Exemple n°2
Début
Riri ← "Loulou"
Fifi ← Riri
Fin
```

La seule différence entre les deux algorithmes consiste dans la présence ou dans l'absence des guillemets lors de la seconde affectation. Et l'on voit que cela change tout !
Dans l'exemple n°1, ce que l'on affecte à la variable Fifi, c'est la suite de caractères R – i – r - i. Et à la fin de l'algorithme, le contenu de la variable Fifi est donc « Riri ».

Dans l'exemple n°2, en revanche, Riri étant dépourvu de guillemets, n'est pas considéré comme une suite de caractères, mais comme un nom de variable. Le sens de la ligne devient donc : « affecte à la variable Fifi le contenu de la variable Riri ». A la fin de l'algorithme n°2, la valeur de la variable Fifi est donc « Loulou ». Ici, l'oubli des guillemets conduit certes à un résultat, mais à un résultat différent.
A noter, car c'est un cas très fréquent, que généralement, lorsqu'on oublie les guillemets lors d'une affectation de chaîne, ce qui se trouve à droite du signe d'affectation ne correspond à aucune variable précédemment déclarée et affectée. Dans ce cas, l'oubli des guillemets se solde immédiatement par une erreur d'exécution.

Ordre des instructions

Il va de soi que l'ordre dans lequel les instructions sont écrites va jouer un rôle essentiel dans le résultat final.
Considérons les deux algorithmes suivants :

```
Exemple 1
Variable A en Numérique
Début
A ← 34
A ← 12
Fin

Exemple 2
Variable A en Numérique
Début
A ← 12
A ← 34
Fin
```

Exercices sur les variables

Exercice 1.1

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 1
B ← A + 3
A ← 3
Fin
```

Réponse :

Exercice 1.2

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C en Entier
Début
A ← 5
B ← 3
C ← A + B
A ← 2
C ← B - A
Fin
```

Réponse :

Exercice 1.3

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 5
B ← A + 4
A ← A + 1
B ← A - 4
Fin
```

Réponse :

Exercice 1.4

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Variables A, B, C en Entier
Début
 A ← 3
B ← 10
C ← A + B
B ← A + B
A ← C
Fin
```

Réponse :

Exercice 1.5

Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
Variables A, B en Entier
Début
A ← 5
B ← 2
A ← B
B ← A
Fin
```

Réponse :

Les deux dernières instructions permettent-elles d'échanger les deux valeurs de B et A ? Si l'on inverse les deux dernières instructions, cela change-t-il quelque chose ?

Exercice 1.6

Ecrire un algorithme permettant d'échanger les valeurs de deux variables A et B, et ce quel que soit leur contenu préalable.

Réponse :

Exercice 1.7

On dispose de trois variables A, B et C. Ecrivez un algorithme transférant à B la valeur de A, à C la valeur de B et à A la valeur de C (toujours quels que soient les contenus préalables de ces variables).

Réponse :

Exercice 1.8

Que produit l'algorithme suivant ?

```
Variables A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A + B
Fin
```

Réponse :

Expressions et opérateurs

Une expression est un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur

voyons quelques expressions de type **numérique**. Ainsi :

```
7

5+4

123-45+844

Toto-12+5-Riri
```

…sont toutes des expressions valides, pour peu que Toto et Riri soient bien des nombres. Car dans le cas contraire, la quatrième expression n'a pas de sens. En l'occurrence, les opérateurs employés sont l'addition (+) et la soustraction (-).

Un opérateur est un signe qui relie deux valeurs, pour produire un résultat.

Les opérateurs possibles dépendent du type des valeurs utilisés.

Opérateurs numériques

Ce sont les quatre opérations arithmétiques tout ce qu'il y a de classique.
+ : addition
- : soustraction
*** : multiplication**
/ : division
Mentionnons également le ^ qui signifie « puissance ». 45 au carré s'écrira donc 45 ^ 2.
Enfin, on a le droit d'utiliser les parenthèses, avec les mêmes règles qu'en mathématiques. La multiplication et la division ont « naturellement » priorité sur l'addition et la soustraction. Les parenthèses ne sont ainsi utiles que pour modifier cette priorité naturelle.
Cela signifie qu'en informatique, 12 * 3 + 5 et (12 * 3) + 5 valent strictement la même chose, à savoir 41.

Opérateur alphanumérique : &

Cet opérateur permet de **concaténer**, autrement dit d'agglomérer, deux chaînes de caractères. Par exemple :

```
Variables A, B, C en Caractère

Début

A ← "Gloubi"

B ← "Boulga"

C ← A & B

Fin
```

La valeur de C à la fin de l'algorithme est "GloubiBoulga"

Opérateurs logiques (ou booléens) :

Il s'agit du ET, du OU, du NON . Nous en parlerons plus tard.

Exercices sur les expressions et operateurs

Exercice 1.8

Que produit l'algorithme suivant ?

```
Variables A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A + B
Fin
```

Exercice 1.9

Que produit l'algorithme suivant ?

```
Variables A, B, C en Caractères
Début
A ← "423"
B ← "12"
C ← A & B
Fin
```

Dernière remarque sur les variables :

En informatique, une variable possède à un moment donné une valeur et une seule.

On parle de l'**état** de la variable

A la rigueur, elle peut ne pas avoir de valeur du tout (une fois qu'elle a été déclarée, et tant qu'on ne l'a pas affectée)