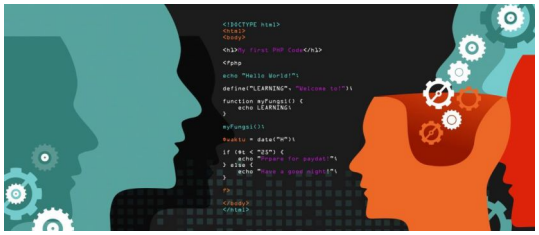
Algorithmique et structure de données



1ère année Tronc commun "Mathématiques et Informatique"

Dr. GAOUAR Lamia

1.0

Février 2021

Table des matières

I - Chapitre 2 : L'affectation **3**

1. Un monde de variables 3

1.1. Définition 3

1.2. Déclaration d'une variable 4

2. L'instruction d'affectation 6

2.1. Syntaxe et signification 6

2.2. Expressions et opérateurs 6

I Chapitre 2 : L'affectation

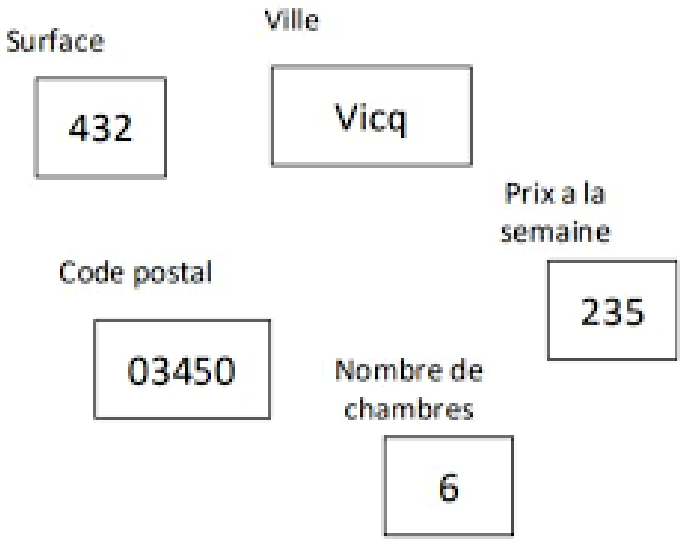
Dans ce chapitre, nous allons nous consacrer à l'étude de la première instruction élémentaire en algorithmique : l'affectation. Nous verrons, sa signification, sa syntaxe et son utilisation. Pour cela, il est indispensable de commencer par présenter ce que sont les variables !

1. Un monde de variables

Dans cette section, nous allons apprendre à stocker les données utiles à l'exécution d'un programme informatique grâce aux variables !

1.1. Définition

Dans un programme informatique, il y a en permanence besoin de stocker provisoirement des informations. Celles-ci peuvent être issues du disque dur, entrées par l'utilisateur au clavier ou encore résultantes de l'exécution du programme. Ces données peuvent être de plusieurs types : des nombres, du texte, etc. Lorsque l'on a besoin de stocker une information au cours d'un programme, on utilise ce qu'on appelle ***une variable*** !



Représentation de variables de différents types

Pour employer une image, une variable est une boîte, que le programme, et donc l'ordinateur, va repérer par ***une étiquette*** qui sera le nom de la variable. Et comme mentionné plus haut, chaque variable possède ***un type***. Pour avoir accès au contenu de la boîte (à ***la valeur de la variable***), il suffit de la désigner par son étiquette.

1.2. Déclaration d'une variable

Avant de pouvoir utiliser une variable vous devez d'abord la déclarer ! Cette opération s'appelle la déclaration des variables. Elle consiste à attribuer un nom et un type à la variable déclarée, et se fait tout au début de l'algorithme, avant toute instruction, dans la partie "déclarations".

En voici la syntaxe générale :

```
1 Var
2 nom_de_variable : type_de_variable;
```

1.2.1. Nommer une variable

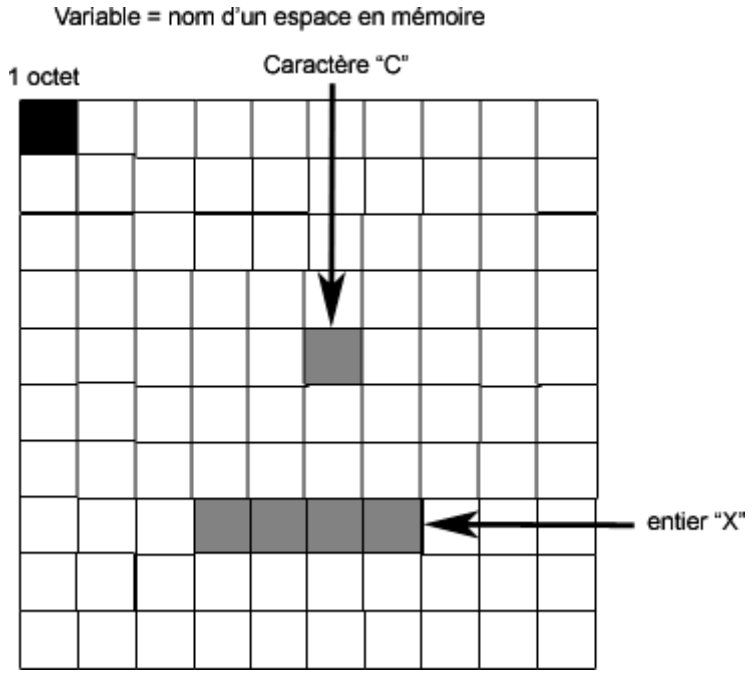
Le nom de la variable obéit à des conventions changeantes selon les langages. Toutefois, certaines règles doivent être suivies, impérativement, quel que soit le langage. Donc, un nom de variable :

- peut comporter des lettres et des chiffres ;
- exclut la plupart des signes de ponctuation, en particulier les espaces ;
- commence par une lettre ;
- et enfin, le nombre maximal de signes pour un nom de variable dépend du langage utilisé.

`nombre_voiture`, `valeur1`, `message`, sont des exemples de nom de variables valides. Par contre : `nombre habitant`, `5moto`, sont des exemples absolument erronés.

1.2.2. Typer une variable

Lorsqu'on déclare une variable, on dit qu'on lui réserve un emplacement mémoire. Faudrait-il encore préciser ce que l'on veut mettre dedans, car de cela dépend la taille de l'emplacement mémoire qui sera alloué à la variable.



Représentation de variables dans la mémoire

Donc, en déterminant le type de la variable, nous allouons une certaine quantité de mémoire destinée à stocker la valeur de la variable. Nous allons tout de suite étudier les différents types de variables.

a) Les variables numériques

Une variable de type numérique est destinée à recevoir des nombres, et uniquement des nombres. Ces nombres peuvent être des entiers ou des réels, de valeur négative ou positive.

Tous les langages, quels qu'ils soient offrent différentes catégories (sous-types) de type numérique, qui sont susceptibles de varier légèrement d'un langage à l'autre. En générale et presque dans tous les langages, on retrouve cependant les types suivants :

Type numérique	Plage de valeurs
Byte	0 à 255
Entier simpe	-32 768 à 32 767
Entier long	-2 147 483 684 à 2 147 483 647
Réel simple	-3,40x10^{38} à -1,40x10^{45} pour les valeurs négatives 1,40x10^{-45} à 3,40x10^{38} pour les valeurs positives
Réel long	1,79x10308 à -4,94x10-324 pour les valeurs négatives 4,94x10-324 à 1,79x10308 pour les valeurs positives

Différents types numériques

La colonne Plage, détermine la plage de valeurs (l'intervalle des valeurs possible) que peut prendre une variable du type correspondant. Par exemple : une variable de type Byte peut prendre les valeurs comprises entre 0 et 255.

En pseudo-code, une déclaration de variables de type numérique se fait comme ceci :

```
1 Var
2 n: Entier;
3 a: réel;
```

b) Les variables en caractères

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. Une variable de type caractères va nous permettre de stocker un seul caractère ou une chaîne de caractères (ou une phrase pour employer un terme plus familier).

En pseudo-code, une déclaration de variables de type caractère se fait comme ceci :

```
1 Var
2 C: char;            //Pour stocker un caractère
3 phrase: char[5];    //Pour stocker une chaîne de 5 caractères max.
```

c) Les variables booléennes

Le dernier type de variables est le type booléen. On y stocke uniquement les valeurs logiques VRAI et FAUX. On peut représenter ces notions abstraites de VRAI et de FAUX par tout ce qu'on veut : de l'anglais (*true* et *false*) ou des nombres (0 et 1).

En pseudo-code, une variable booléenne se déclare comme suit :

```
1 Var
```

```
b: bool;
```

2. L'instruction d'affectation

Pour faire le point : une variable est caractérisée par un nom, un type et une valeur. Après avoir vu comment déclarer une variable, et par conséquent définir son nom et son type, nous allons voir comment lui attribuer une valeur. En algorithmique, cette opération s'appelle l'affectation.

2.1. Syntaxe et signification

En pseudo-code, l'instruction d'affectation se note avec le signe **:=**

Ainsi :

```
t := 24;
```

Attribue la valeur 24 à la variable t. Ceci sous-entend impérativement que t soit une variable de type entier. Si t a été défini dans un autre type, il faut bien comprendre que l'instruction précédente aurait provoqué une erreur.

Voici un autre exemple :

```
r := "salut lili";
```

Attribue la valeur "salut lili" à la variable r. Celle-ci encore sous-entend impérativement que r soit une variable de type caractères et qu'elle ne peut contenir que ça. Notez la présence OBLIGATOIRE des guillemets qui entourent la chaîne de caractères.

2.2. Expressions et opérateurs

Dans une instruction d'affectation, on trouve :

- à gauche de :=, un nom de variable, et uniquement cela ;
- à droite de :=, ce qu'on appelle une expression. En informatique, le terme d'expression désigne un ensemble de valeurs, reliées par des opérateurs, et équivalent à une seule valeur.

Ça paraît obscur pour l'instant, mais vous allez vite comprendre. Par exemple, observons ces quelques expressions de type numérique :

```
a := 7;
b := 5 + 4;
c := 123 - 45 + 844;
d := toto - 12 + 5 + riri;
```

Ainsi ces exemples sont toutes des expressions valides. Cependant, pour la dernière expression, toto et riri sont obligatoirement des nombres. Car dans le cas contraire, cela n'aurai pas de sens. En l'occurrence, les opérateurs employés sont l'addition (+) et la soustraction (-). Or, on ne peut additionner et soustraire que des valeurs numériques !

Dans une opération d'affectation, l'expression située à droite de l'opération doit être du même type que la variable située à gauche. a, b, c et d sont donc supposés être des variables numériques.

J'attire maintenant votre attention sur le terme opérateur : un opérateur est un signe qui relie deux valeurs, pour produire un résultat. Les opérateurs possibles dépendent du type des valeurs qu'on manipule dans l'expression. Nous allons de suite, voir les différents types d'opérateurs que vous pouvez rencontrer :

2.2.1. Opérateurs numériques

Ce sont les quatre opérations arithmétiques que vous connaissez tous :

- « + » : l'addition ;
- « - » : la soustraction ;
- « * » : la multiplication ;
- « / » : la division ;
- La puissance quant à elle, est représentée par le signe « ^ ». Par exemple : 45 au carré s'écrira donc 45 ^ 2.

Enfin, la priorité des opérateurs reste la même, à savoir, la multiplication et la division sont prioritaires sur l'addition et la soustraction. Et pour modifier cette priorité, utilisez les parenthèses avec les mêmes règles qu'en mathématiques.

2.2.2. Opérateur de concaténation

Cet opérateur permet de concaténer, autrement dit de coller, deux chaînes de caractères. Par exemple :

```
1 Var
2 a, b, c: char[10];
3 Debut
4 a := 'nou';
5 b := 'nours';
6 c := a+b;
7 Fin.
```

La valeur de c à la fin de l'algorithme est 'nounours'.

2.2.3. Opérateurs logiques (ou booléens)

Il s'agit du ET, du OU, du NON et du XOR. Nous les laisserons de côté provisoirement, nous y reviendrons plus tard.