# VARIABLES ET CONSTANTES ET TYPES DE DONNEES EN C

Une variable en termes simples est un lieu de stockage auquel de la mémoire est allouée. Fondamentalement, une variable utilisée pour stocker une forme de données. Différents types de variables nécessitent différentes quantités de mémoire et disposent d'un ensemble d'opérations spécifique qui peut leur être appliqué.

## Déclaration

```
type nom_variable;

//ou pour plusieurs variables:
type nom-variable1, nom-variable2, nom-variable3;
```

**Remarque :** Un nom de variable peut être composé d'alphabets (en majuscules et minuscules), de nombres et du caractère de soulignement "_". Toutefois, le nom ne doit pas commencer par un chiffre.

**Exemple 1 :**

```
1    int a;
2    float b=1, c;
```

**Règles de nommer des identificateurs de variables**

- Un identificateur valide peut avoir des lettres (en majuscules et en minuscules), des chiffres et des soulignements.
- La première lettre d'un identificateur doit être une lettre ou un trait de soulignement.
- Vous ne pouvez pas utiliser des mots-clés comme identificateurs.
- Il n'y a pas de règle sur la longueur d'un identificateurs. Cependant, certains compilateurs peuvent rencontrer des problèmes si l'identificateur comporte plus de 31 caractères.

Vous devriez toujours essayer de donner des noms significatifs aux variables. Par exemple: "**Age**" est un meilleur nom de variable que "**a**".

C est un langage fortement typé. Cela signifie que le type de variable ne peut pas être modifié une fois déclaré. Par exemple:

**Exemple 2 :**

```
1    int a;
2    a=5.6 // Erreur
3    float a; //Erreur
```

Ici, le type de "a" int. Vous ne pouvez pas attribuer une valeur à virgule flottante (décimale) 5.5 à cette variable. En outre, vous ne pouvez pas redéfinir le type de

données de la variable. Par ailleurs, pour stocker les valeurs décimales en C, vous devez déclarer son type à double ou float.

## Constantes

Si vous voulez définir une variable dont la valeur ne peut pas être changée, vous pouvez utiliser le mot-clé const. Cela va créer une constante. Exemple,

**Exemple 3 :**

```
1    const double PI=3.14;
```

Ici, PI est une constante; sa valeur ne peut être changée.

**Exemple 4 :**

```
1    const double PI=3.14;
2    PI=4.6; // Erreur
```

Vous pouvez également définir une constante en utilisant #define.

# TYPES DE DONNÉES EN C

Chaque variable en C a un type de données associé. Chaque type de données nécessite différentes quantités de mémoire et a quelques opérations spécifiques qui peuvent être effectuées sur elle.

## Types de base

- **char**: Le type de données le plus fondamental en C. Il stocke un seul caractère et nécessite un seul octet de mémoire dans presque tous les compilateurs.
- **int**: une variable int est utilisée pour stocker un entier.
- **float**: Il est utilisé pour stocker des nombres décimaux (nombres avec une valeur à virgule flottante) avec une simple précision.
- **double**: Il est utilisé pour stocker des nombres décimaux (nombres avec une valeur à virgule flottante) avec une double précision.

Différents types de données ont également différentes plages dans lesquelles ils peuvent stocker des nombres. Ces plages peuvent varier d'un compilateur à l'autre. Vous trouverez ci-dessous une liste de plages ainsi que les exigences en matière de mémoire et de format sur un compilateur gcc 32 bits.

| Type de données | Taille (octets) | plage de valeurs | Spécificateur de Format |
|---|---|---|---|
| short int | 2 | -32,768 to 32,767 | %hd |
| unsigned short int | 2 | 0 to 65,535 | %hu |
| unsigned int | 4 | 0 to 4,294,967,295 | %u |
| int | 4 | -2,147,483,648 to 2,147,483,647 | %d |
| long int | 4 | -2,147,483,648 to 2,147,483,647 | %ld |
| unsigned long int | 4 | 0 to 4,294,967,295 | %lu |
| long long int | 8 | -(2^63) to (2^63)-1 | %lld |
| unsigned long long int | 8 | 0 to 18,446,744,073,709,551,615 | %llu |
| signed char | 1 | -128 to 127 | %c |
| unsigned char | 1 | 0 to 255 | %c |
| float | 4 | | %f |
| double | 8 | | %lf |
| long double | 12 | | %Lf |

#### void

**void** est un type incomplet. Cela signifie "rien" ou "pas de type". Vous pouvez penser à void comme absent.
Par exemple, si une fonction ne renvoie rien, son type de retour doit être nul.
Notez que vous ne pouvez pas créer de variables de type **void**.

Nous pouvons utiliser l'opérateur sizeof() pour vérifier la taille d'une variable.

**Exemple 1 :**

```
1
2       #include < stdio.h>
3
4       int main(void)
5       {
6           int a=2;
7
8           // printf("la taille de a est %lu octets",sizeof(int));
9           printf("la taille de a est %lu octets",sizeof(a));
10
11          return;
        }
```

```
la taille de a est 4 octets
```