

# LES NOTIONS DE BASE

## I. LES IDENTIFICATEURS

Les identificateurs servent à désigner par un **nom unique** les différents objets manipulés par le programme. Un identificateur peut désigner :

- Un Nom de Variable ou de fonction,
- Un nouveau Type désigné par le programmeur, - Une étiquette.

**NB :** Un identificateur doit être composée des caractères suivants

- Des lettres Minuscule et ou Majuscule Non-Accentuées
- Des Chiffres
- Le tiret du Huit ou BLANC SOULIGNE ( \_ )

**L'identificateur ne doit commencer par un CHIFFRE.**

**EX :** x, y1, y2, AB, ...

## EXERCICE

Parmi ces identificateurs dites ceux qui sont corrects ou pas.

Aly Tall NIANG		ma variable		const	
_aly		V@R		école	
\$12a		AaAaAa		ecole	
65-6		_BA_		a,b,c	
Tous_les_hommes		DuDu7		papa	

## II. LES TYPES DE BASE

Tous les objets manipulés (**Variables, Constantes, Fonctions**) doivent avoir un type bien défini : Le **LANGAGE C est un Langage TYPE**.

Le type d'un objet détermine la manière dont il sera représenté en mémoire :

- ✦ La **PLAGE DE VALEUR** de L'objet.
- ✦ **L'ESPACE DE RÉSERVATION EN MÉMOIRE.**
- ✦ La **TAILLE MÉMOIRE** Correspondant.

La taille mémoire des différents types dépend des compilateurs.

### **a) Les Caractères**

Ils sont désignés par le mot clé « **char** ». Un objet de type '**char**' peut contenir n'importe quel symbole du jeu de caractère du clavier utilisé. C'est l'objet le plus élémentaire en C : il est codé sur un **OCTET**. Tous les caractères peuvent être représenté en entier.

**Ex :** Voir le tableau des Codes **ASCII**.

## b) Les Entiers

Ils représentent l'ensemble  $\mathbb{N}$  et sont représenté par le mot clé **int**. Cet ensemble est divisé en trois sous-ensembles selon la taille de l'entier : **short int** (ou short), **int** et **long int** (ou long). Chacun de ces trois types peut être nuancé par l'utilisation du qualificatif **unsigned** (non signé), dans ce cas, il n'y a plus de bit réservé au signe et on ne peut représenter que les entiers positifs.

	Taille	Désignation
<b>char</b>	32 bits	Flottant
<b>short</b>	64 bits	Flottant double précision
<b>int</b>	128 bits	Flottant quadruple précision
<b>Long</b>	32 bits	Entier long
<b>Long long</b>	64 bits	Entier long

## c) Les Flottants

Ils permettent de représenter une partie de nombres réels. Ils peuvent être représentés par les mots clés **float**, **double** et **long double**.

	Taille	Désignation
<b>Float</b>	32 bits	Flottant
<b>Double</b>	64 bits	Flottant double précision
<b>Long double</b>	128 bits	Flottant quadruple précision

**Remarque :** pour connaître la taille d'un type ou d'un objet, on utilise la fonction **sizeof(expression)**. La fonction donne le résultat qui est un entier représentant le nombre d'octets nécessaires pour stocker le type ou l'objet.

**Exemple :** (à noter)

## d) Les Constantes

Une constante est une valeur qui ne change pas durant tout le long du programme et qui apparaît littéralement dans le code source d'un programme, le type de la constante étant

déterminé par la façon dont la constante est écrite. Les constantes peuvent être de 4 types : entier, flottant (nombre réel), caractère, énumération.

### DECLARATION ET INITIALISATION DE CONSTANTES

Pour définir une constante, sa déclaration doit se faire en même temps que son initialisation.

Syntaxe : (à noter)

## e) Les Variables

Les variables sont au cœur de la programmation impérative. On a besoin de variable dans tous les programmes. Elles associent un nom (identificateur) à une valeur. Une variable, comme son nom l'indique, peut changer de valeur dans le même programme.

### DECLARATION ET INITIALISATION DE VARIABLES

Syntaxe :

```
type nom_variable ;  
type nom_variable1, nom_variable2,... ;
```

*L'initialisation d'une variable peut se faire pendant ou après sa déclaration.*

Exemple : `int a ; | int a =2 ;`

NB :

*Le nom de la variable dépend du programmeur. Cependant, il est recommandé de choisir des noms courts et explicites (le nom d'une variable est en rapport avec son contenu).*

## III. LES OPERATEURS

### 1) Les Opérateurs d'Affectation

Ce type d'opérateur permet de modifier la valeur d'une variable. Il est symbolisé par le signe =

Syntaxe : `variable = expression`

Le terme de gauche reçoit la valeur après évaluation de l'expression de droite.

Exemples : (à noter)

### 2) Les Opérateurs Arithmétiques

Les opérateurs arithmétiques sont : **+** (*addition*), **-** (*soustraction*), **\*** (*multiplication*), **/** (*division*) et **%** (*modulo*). Ce dernier permet de retourner le reste de la division, **il est seulement utilisé pour des entiers**.

### 3) Les Opérateurs d'Accumulation

Ils permettent de modifier la valeur d'une variable : **+=** (*addition*), **-=** (*soustraction*), **\*=** (*multiplication*), **/=** (*division*) et **%=** (*modulo*).

Exemples : (à noter)

### 4) Les Opérateurs d'Incrément et de Décrément

Les opérateurs d'**incrément** « **++** » et de **décrément** « **--** » s'utilisent aussi bien en suffixe (*post-incrément*) qu'en préfixe (*pré-incrément*). Dans les deux cas la variable *i* sera incrémenté, toutefois dans la notation suffixe la valeur retournée sera l'ancienne valeur de *i* alors que dans la notation préfixe se sera la nouvelle.

Exemples : (à noter)

### 5) Les Opérateurs Relationnels ou de Comparaison

Ils sont en général utilisés dans les conditions :

- ✦ **>** Strictement supérieur
- ✦ **<** Strictement inférieur
- ✦ **>=** supérieur ou égal
- ✦ **<=** inférieur ou égal
- ✦ **==** égal
- ✦ **!=** différent

### 6) Les Opérateurs Logique

⑥ Le ET logique (&&).

Il permet de relier plusieurs conditions avec la règle suivante :

Si une condition globale est composée de plusieurs sous-conditions reliées par des **&&**, alors la condition globale est vérifiée si toutes les sous-conditions sont vérifiées.

### Exemples :

**Table de vérité C1/C2 :** (On multiplie C1 et C2)

C1	C2	C1 && C2

### ○ Le OU logique (||) :

Il permet de relier plusieurs conditions avec la règle suivante : si une condition globale est composée de plusieurs sous-conditions reliées par des **||**, alors la condition globale est vérifiée si une des sous-conditions est vérifiée.

**Exemples :** (à noter)

**Table de vérité C1/C2 :** (On additionne C1 et C2)

C1	C2	C1    C2

## **7) L'Opérateur Virgule**

Une expression peut être constituée d'une suite d'expressions séparées par des virgules : **expression-1, expression-2, ... , expression-n**

Cette expression est alors évaluée de gauche à droite. Sa valeur sera la valeur de l'expression de droite. Par exemple :

**X = ((a =3), (a++), (a+6)) => x = 10**

## IV. LES TYPES DE BASE

### 1) La Sortie printf()

En langage C, pour afficher des informations à l'écran, on utilise la fonction **printf ()**.

Il existe plusieurs types d'affichage :

- **Affichage de message sans valeur**

Ce type d'affichage est utilisé pour donner des informations (message simple) à

```
printf(« Hello world ! »)
```

L'utilisateur. Par exemple ;

- **Affichage de message avec valeurs**

Cette méthode est utilisée s'il y'a des valeurs qu'on veut afficher.

Par exemple :

```
printf(« code format », variable);
```

Il y aura autant de code format que de valeur à afficher (exemple

```
printf("S=%d+%d=%d",x,y,x+y) ); 😊
```

### Tableau des codes format

(à noter)

**Exemple :** (à noter)

### 2) La Sortie scanf()

Cette fonction permet de saisir dans une variable.

Syntaxe :

```
Scanf(« code formats », &variable);
```

Il y aura autant de codes format que de variables séparées par des virgules.

## STRUCTURE D'UN PROGRAMME EN C

```
[Directives au préprocesseur]  
[Déclarations de variables externes]  
[Fonctions secondaires]  
Main ()  
{  
Déclarations de variables internes  
Instructions  
}
```

### Applications :

- Ecrire un programme qui permet d'effectuer la somme de deux variables entières saisies au clavier.
- Ecrire un programme qui permet de saisir une longueur et une largeur d'un rectangle puis affiche : le périmètre, le demi-périmètre et la surface du rectangle.