

## Langage C : résumé

### 1 Syntaxe du langage C

Les **identifiants** permettent de donner un nom à un élément du programme (variable, fonction, ...).

Une **instruction** est une expression ou une suite d'expressions qui se termine par le caractère ; .

Une suite d'expressions peut s'écrire en utilisant le séparateur , .

Les **blocs** d'instructions sont délimités avec les caractères { et } .

Une **déclaration** permet d'indiquer le type de donnée d'une variable.

```
int x; // Un entier x
double y, z; // Deux réels y et z
```

L'**initialisation** permet de donner une valeur à une variable :

```
z=4.78e-4; //4,78.10-4
```

La déclaration et l'initialisation peuvent se faire en même temps.

```
double y=3.9, z;
```

### 2 Les Commentaires

Sur une ligne : // Voici un commentaire sur une seule ligne.

Un bloc de commentaires : /\* Ceci est un commentaire sur  
plusieurs lignes \*/

### 3 Les Types de Données

Nous avons :

- . Les caractères : **char**.  

```
char ex='c';
```
- . Les entiers : **short**, **int**, **long**, **long long**.
- . Les réels : **float**, **double**, **long double**.

Les entiers sont signés par défaut, mais il est possible de préciser des valeurs strictement positives à l'aide du mot clé **unsigned**.

Par exemple, **short** permet de sélectionner un nombre entier compris entre  $[-32768; +32767]$ , alors que **unsigned short** lui autorise un nombre entier compris entre  $[0; 65535]$ .

### 4 Les Opérateurs

Il y en a plusieurs :

- . Les opérateurs arithmétiques : +, -, \*, / et %.
- . Les opérateurs d'assignation : =, +=, -=, \*=, /= et %=.  

```
x=x+2; revient à x+=2;
```
- . Les opérateurs de comparaison : ==, <, <=, >, >=, et !=.
- . Les opérateurs logiques : ||, &&, !.
- . Les opérateurs d'incrément : ++, --.  

```
x=x+1; revient à x++;
```

## 5 Les Entrées-Sorties

Ce sont des saisies ou des affichages formatés de données.

Pour l'affichage à l'écran, il faut inclure le fichier `stdio.h` (STandarD Input Output) et mettre dans un bloc l'instruction suivante :

```
printf("format",var1,var2,... );
```

Par exemple,

```
printf("Valeurs : %d,%d,%d\n",10,15,20); // Affiche à l'écran Valeurs : 10,15,20
```

Pour la lecture au clavier, il faut inclure le fichier `stdio.h` (STandarD Input Output) et mettre dans un bloc l'instruction suivante :

```
scanf("format",&var1,&var2,... );
```

Par exemple,

```
scanf("%d",&nombreEtudiants);
```

Attention, si une variable `nomEtudiant` est de type chaîne de caractères, il faut écrire :

```
scanf("%s",nomEtudiant);
```

Les principaux formats sont :

%d	int (nombre décimal)
%ld	long (nombre long décimal)
%u	unsigned int (nombre décimal non signé)
%x	int (nombre hexadécimal)
%f	double (virgule fixe)
%lf	long double (virgule fixe)
%e	double (notation exponentielle)
%c	caractère
%s	chaîne de caractères

## 6 Les Structures Conditionnelles

### 6.1 Structure if/else

Si la condition est vraie, nous exécutons le bloc 1 d'instructions sinon le bloc 2 d'instructions.

```
int x=5, y;
if (x==2) // x est égal à 2?
{ // Bloc 1
    y=2;
}
else // x est différent de 2!!!
{ // Bloc 2
    y=x+2;
}
```

### 6.2 Structure switch

Les instructions sont exécutées suivant la valeur de la variable testée.

L'instruction `break` termine une série d'instructions.

```
int x=5, y;
switch (x)
{
    case 1 : y=3; break; //Si x=1 alors y=3
    case 2 : case 3 : y=x+6; break; // Si x=2, y=8 et si x=3, y=9
    default : y=x; break; // Sinon
}
```

## 7 Les Boucles

Elles permettent d'exécuter plusieurs fois un même bloc d'instructions.

Il faut définir une expression de test de fin de boucle.

L'instruction **break** peut interrompre la boucle.

L'instruction **continue** permet de ne pas prendre en compte la suite des instructions jusqu'au test de fin de boucle, soit d'arrêter l'itération courante et de passer au début de l'itération suivante.

### 7.1 Boucle while

Tant que la condition est vraie, nous exécutons le bloc d'instructions.

```
int x=0; // Initialisation
while (x<5) // Test de fin de boucle
{
    printf("x=%d\n",x);
    x++; // Valeur suivante
}
```

### 7.2 Boucle do/while

Le test de fin de boucle est évalué après le bloc.

```
int x=0; // Initialisation
do
{
    printf("x=%d\n",x);
    x++; // Valeur suivante
}
while (x<5); // Test de fin de boucle
```

### 7.3 Boucle for

Il y a trois instructions : initialisation, test de fin de boucle et valeur suivante.

La syntaxe avec définition de x dans le for est la suivante :

```
for (int x=0; x<5; x++)
{
    printf("x=%d\n",x);
}
```

## 8 Les Fonctions

La **fonction principale** est le point d'entrée d'un programme C.

Elle possède d'ailleurs plusieurs syntaxes :

```
int main();
int main(void);
int main(int argc, char **argv);
```

Les **fonctions** permettent de réaliser un traitement particulier (réutilisable) dans un bloc d'instructions spécifique.

Un **prototype** permet de déclarer une fonction (dans les fichiers .h) en indiquant obligatoirement :

- Le type de retour (**void** s'il n'y a pas de valeur de retour);
- Le nom de la fonction;
- Les arguments (ou paramètres) de la fonction entre parenthèses.

Par exemple,

```
double calcul(int a,double b);  
void affiche();
```

La valeur de retour est donnée (si besoin) à l'aide du mot clé **return**.

Par exemple,

```
double calcul(int a,double b)  
{  
    if (a>0) return a+b;  
    return b-a;  
}
```

Vous pouvez alors utiliser cette fonction comme ceci :

```
double x,y =calcul(5,3.6);  
x=calcul(-5,6.254);
```

## 9 Valeur pointée

Pour obtenir la valeur pointée (le contenu de l'adresse mémoire ou le contenu du pointeur), nous utilisons à nouveau le caractère **\***.

```
int x=4;  
int *pt=&x;  
printf("Adresse de x : %p\n",pt);  
printf("Valeur de x : %d\n",*pt);
```

## 10 Les Tableaux

Les tableaux sont des pointeurs constants (adresse mémoire non modifiable).

Nous utilisons les caractères `[ et ]` pour la définition et la récupération de valeur.

Nous pouvons utiliser les caractères `{ et }` pour l'initialisation.

```
int tab[30]; // Tableau de 30 entiers  
int tab2[5]={4,5,2,1,3};  
int x=tab2[3]; // 1 (4ème case)
```

## 11 Les Variables Globales

Les variables globales sont définies en dehors de toute fonction ou notées **static** :

```
int a=22; // Variable globale  
...  
int main()  
{  
    float b=5.2; // Variable locale  
    static int c=8; // Variable globale  
    ...  
}
```