%M….

**Chapitre 3. Les entrées-sorties conversationnelles**

La bibliothèque standard contient un ensemble de fonctions d’entrées-sorties conversationnelles (lecture au clavier, affichage à l'écran).

# Fonction printf()

La fonction **printf()** permet d’afficher du texte, des valeurs de variables ou des résultats d'expressions à l'écran.

**Syntaxe** : printf(format[,arg1, arg2,…,argn]).

* + Le paramètre **format** désigne une chaîne de caractères comprenant :

1. des caractères à afficher tels quels,
2. des caractères non imprimables (séquences d'échappement),
3. des spécificateurs de format : suite de caractères précédée du symbole % précisant la manière dont les valeurs des arguments <arg1..n> sont affichées.

Une spécification de format est de la forme :

% [drapeau] [largeur] [.précision] [modificateur] code\_conversion.

* + Les arguments à afficher peuvent être des constantes littérales ou des variables ou des expressions.

**Note** : Il doit y avoir autant de spécificateurs de format que d’arguments.

## Le champ code de conversion

Le format minimum d’impression est constitué du caractère % suivi d’un code de conversion qui spécifie le type de donnée de l’argument à afficher, dont voici les principaux :

|  |  |  |
| --- | --- | --- |
| **Code de conversion** | **Type** | **Remarque** |
| d ou i | int |  |
| u | unsigned int |  |
| o | int | affiché en octal |
| x ou X | int | affiché en hexadécimale |
| c | char |  |
| f | float ou double |  |
| e ou E | float ou double | affiché en notation scientifique |
| s | char\* |  |

## Exemple 1 :

int q = 100; float p = 60.50; char t =’A’;

printf("Quantite = %d\n",q); printf("Prix = %f\n", p) ; printf("Taille = %c\n",t) ; printf("Prix total = %f", q\*p);

### ou bien

printf("Quantite = %d\nPrix = %f\nTaille = %c\nPrix total = %f",q,p,t,q\*p);

### Imprime

Quantite = 100

Prix = 60.500000

Taille = A

Prix total = 6050.000000

## Le champ modificateur

Le champ modificateur est optionnel, il est constitué d’une simple lettre (l, h ou L) placée devant le champ code de conversion qui spécifie **short** ou **long.**

Exemples des combinaisons typiques des champs modificateurs et codes de conversion :

ld ou li : **long int**.

lu : **unsigned long int.**

hd : **short int**.

Lf ou Le : **long double**.

## Spécificateur de précision

Par défaut, les flottants sont affichés avec six chiffres après le point décimal (aussi bien pour la notation décimale que pour la notation exponentielle).Il est possible de modifier cette représentation en utilisant un point décimal et un spécificateur de précision.

## Exemple 2 :

float val = 25.1234;

printf("%f",val); */\* affiche : 25.123400 \*/* printf("%.0f",val); */\* affiche : 25 \*/* printf("%.1f",val); */\* affiche : 25.1\*/* printf("%.2f",val); */\* affiche : 25.12\*/*

printf("%e",val); */\* affiche : 2.512340e+01\*/*

printf("%.3e",val); */\* affiche : 2.512e+01\*/*

## Le champ largeur

Le champ *largeur* permet de spécifier le nombre minimal de caractères à afficher.

## Exemple 3 :

int val1 = 125; float val2 = 4.5;

printf("|%d|",val1); */\* affiche : |125| \*/* printf("|%2d|",val1); */\* affiche : |125| \*/* printf("|%5d|",val1); */\* affiche : |--125| \*/* printf("|%05d|",val1); */\* affiche : |00125| \*/*

printf("|%8.2f|\n",val2); */\* affiche : |----4.50| \*/*

## Le champ drapeau

* + - "-" : Cadrage à gauche (sinon à droite par défaut).
    - "+" : affichage du signe pour les nombres positifs.

## Exemple 4 :

int val1 = 125; float val2 = 4.5;

printf("|%-5d|",val1); */\*affiche : |125--|\*/*

printf("|%+05d|",val1); /\*affiche : |+0125|\*/ printf("|%-8.2f|\n",val2); /\*affiche : |4.50 |\*/

# Fonction scanf()

La fonction *scanf* permet de lire des données au clavier et de les affecter à des variables.

**Syntaxe** : scanf(format,arg1, arg2,…,argn). Une spécification de format est de la forme :

%[largeur][modificateur] code\_conversion.

## Remarques :

* + Chaque nom d'argument qui correspond à un scalaire doit être précédé de & (opérateur d’adresse), ceci indique que l'information saisie va être placée à l'adresse de la variable.
  + La chaîne de format ne doit comporter que des spécifications de format, sinon tout autre caractère indiqué dans la chaîne de format doit être saisi par l’utilisateur.
  + Le code de conversion **"f"** s’applique aux données de type **float et double** dans le cas de la fonction printf(). Par contre, pour la fonction scanf(), une distinction s’impose : **"f"** ne s’applique qu’aux données de type **float** ; pour les données de type **double**, il faut utiliser le code de conversion **"lf"** qui signifie **long float.**
  + Le champ largeur change par rapport à printf. Il permet de spécifier le nombre maximal de caractères à lire.

## Exemples :

* + Saisie d'une variable

{

float rayon, perimetre;

**printf**("Donnez le Rayon : ");

*/\* Saisie du Rayon \*/*

**scanf**("%f",**&**rayon);

*/\* Calcul du périmètre \*/*

perimetre = 2\*3.14\*rayon;

*/\* Affichage de périmètre \*/*

**printf**("Le périmètre = %.2f", perimetre);

}

* + Saisie de plusieurs variables.

{

float a, b, c, det ;

**printf**("Donnez les valeurs de a,b et c : ");

*/\* Saisie de a, b, c \*/*

**scanf**("%f %f %f",**&**a,**&**b,**&**c);

*/\* Calcul du déterminant \*/*

**det** = b\*b- 4\*a\*c;

**printf**("Le déterminant = %.2f", det);

}

## Tampon de scanf

Les informations tapées au clavier sont d'abord mémorisées dans un emplacement mémoire appelé *tampon* (buffer). Elles sont mises à la disposition de scanf après l’activation de la touche

<Entrée>

* + Pour les nombres : scanf avance jusqu'au premier caractère différent d'un séparateur (espace, tabulation, retour à la ligne) puis scanf prend en compte tous les caractères jusqu'à la rencontre d'un séparateur ou d'un caractère invalide ou en fonction du largeur.
  + Pour les caractères %c : scanf prend le caractère courant séparateur ou non séparateur.

**Exemple** (^ = espace, @ = fin de ligne) :

scanf("%d%d",&a,&b);

123^45@ ==> a = 123 b = 45

10@@20@ ==> a = 10 b = 20

scanf("%d%c",&a,&b);

200^z ==> a = 200 b = ' '

scanf("%d^ %c",&a,&b);

200^z ==> a = 200 b = 'z '

scanf("%d",&a);

15@ ==> a = 15

scanf("%c",&b); ==> b = ‘\n’

# Exemple de programme complet

Le programme suivant calcule la moyenne de deux nombres entrés au clavier et l'affiche :

#include <stdio.h>

void main() *// Point d'entré du programme*

{ */\*Début du programme\*/*

float x, y;

printf("\t\t\tCalcul de la moyenne\n\n"); */\*Affiche le titre\*/*

printf("Entrez le premier nombre : ");

scanf("%f", &x); */\*Lecture du premier nombre\*/*

printf("Entrez le deuxieme nombre : ");

scanf("%f", &y); */\*Lecture du deuxième nombre\*/*

printf("La valeur moyenne de %.2f et de %.2f est %.2f\n",x, y,(x+y)/2);

} *// Fin du programme*

## Commentaire :

Inclusion des fichiers : La première ligne contient la directive **#include** suivi d’un nom de fichier a pour effet d’insérer le fichier spécifié entre **<** et **>**, ici stdio.h, dans le fichier source à l’endroit où la directive est placée. Le fichier d’en-tête **stdio.h** contenant les déclarations nécessaires à l’utilisation des fonctions d’entrées-sorties standard. Le compilateur dispose ainsi des informations nécessaires pour vérifier si l’appel de la fonction (en l'occurrence printf et scanf) est correct.

A propos de main() : Tout programme C doit contenir au moins une fonction appelée main. Le code qu’il contient est encadré par deux accolades, { et }. L’exécution du programme débute immédiatement après l’accolade ouvrante et se termine lorsque l’accolade fermante correspondante est rencontrée.

Ajout de commentaires : Les commentaires permettent de documenter les programmes sources. Ils sont encadrés par " /\*" et par " \*/".

Vous pouvez, en outre, utiliser des « commentaires de fin de ligne » en introduisant les deux caractères : //. Dans ce cas, tout ce qui est situé entre **//** et la fin de la ligne est un commentaire. Les comme

ntaires sont alors ignorés par le compilateur.