

Cours Programmation I (chapitre 3)

Licence Fondamentale SMI (semestre 3)

Pr. Mouad BEN MAMOUN

m.benmamoun@um5r.ac.ma

Année universitaire 2022/2023

Chapitre 3

Entrées-sorties

Les instructions de lecture et d'écriture

- Il s'agit des instructions permettant à la machine de dialoguer avec l'utilisateur
 - Dans un sens la **lecture** permet à l'utilisateur d'entrer des valeurs au clavier pour qu'elles soient utilisées par le programme
 - Dans l'autre sens, **l'écriture** permet au programme de communiquer des valeurs à l'utilisateur en les affichant à l'écran (ou en les écrivant dans un fichier)
- La bibliothèque standard **<stdio>** contient un ensemble de fonctions qui assurent la lecture et l'écriture des données. Dans ce chapitre, nous allons présenter les plus importantes:
 - **printf()** écriture formatée de données
 - **scanf()** lecture formatée de données

Ecriture formatée de données: printf ()

- la fonction **printf** est utilisée pour afficher à l'écran du texte, des valeurs de variables ou des résultats d'expressions.
- Syntaxe : **printf("format", expr1, expr2, ...);**
 - **expr1,...** : sont les variables et les expressions dont les valeurs sont à représenter
 - **format** : est une chaîne de caractères qui peut contenir
 - du texte
 - des séquences d'échappement ('\\n', '\\t', ...)
 - des spécificateurs de format : un ou deux caractères précédés du symbole %, indiquant le format d'affichage

Rq : Le nombre de spécificateurs de format doit être égale au nombre d'expressions!

Spécificateurs de format

SYMBOLE	TYPE	AFFICHAGE COMME
<i>%d ou %i</i>	<i>int</i>	entier relatif
<i>%u</i>	<i>unsigned int</i>	entier naturel non signé
<i>%c</i>	<i>char</i>	caractère
<i>%o</i>	<i>int</i>	entier sous forme octale
<i>%x ou %X</i>	<i>int</i>	entier sous forme hexadécimale
<i>%f</i>	<i>float, double</i>	réel en notation décimale
<i>%e ou %E</i>	<i>float, double</i>	réel en notation exponentielle
<i>%s</i>	<i>char*</i>	chaîne de caractères

Séquences d'échappement

- l'affichage du texte peut être contrôlé à l'aide des *séquences d'échappement* :
 - **\n** : nouvelle ligne
 - **\t** : tabulation horizontale
 - **\a** : signal sonore
 - **\b** : retour arrière
 - **\r** : retour chariot
 - **\v** : tabulation verticale
 - **\f** : saut de page
 - **** : back slash (\)
 - **\'** : apostrophe
 - **\''** : guillemet

Exemples de printf()

```
#include<stdio.h>

main()
{ int i=1 , j=2, N=15;
  printf("la somme de %d et %d est %d \n", i, j, i+j);
  printf(" N= %x \n" , N);
  char c='A' ;
  printf(" le code Ascii de %c est %d \n", c, c);
}
```

Ce programme va afficher :

la somme de 1 et 2 est 3

N=f

le code Ascii de A est 65

Remarque : Pour pouvoir traiter correctement les arguments du type long, il faut utiliser les spécificateurs %ld, %li, %lu, %lo, %lx

Exemples de printf()

```
#include<stdio.h>
main()
{ double x=10.5, y=2.5;
  printf("%f divisé par %f est égal à %f \n", x, y, x/y);
  printf("%e divisé par %e est égal à %e\n", x, y, x/y);
}
```

Ce programme va afficher :

10.500000 divisé par 2.500000 égal à 4.200000

1.050000e+001 divisé par 2.500000e+000 égal à 4.200000e+000

Remarque : Pour pouvoir traiter correctement les arguments du type long double, il faut utiliser les spécificateurs %Lf et %Le

Remarques sur l'affichage

- Par défaut, les entiers sont affichés sans espaces avant ou après
- Pour agir sur l'affichage → un nombre est placé après % et précise le nombre de caractères **minimum à utiliser**

- Exemples : `printf("%4d", n);`

`n = 20` → `~~20` (~ : espace)

`n=56123` → `56123`

`printf("%4X", 123);` → `~~7B`

`printf("%4x", 123);` → `~~7b`

Remarques sur l'affichage

- Pour les réels, on peut préciser la *largeur minimale* de la valeur à afficher et le nombre de chiffres après le point décimal.
- La précision par défaut est fixée à six décimales. Les positions décimales sont arrondies à la valeur la plus proche.
- Exemples :

`printf("%f", 100.123);` ➔ 100.123000

`printf("%12f", 100.123);` ➔ ~~100.123000

`printf("%.2f", 100.123);` ➔ 100.12

`printf("%5.0f", 100.123);` ➔ ~~100

`printf("%10.3f", 100.123);` ➔ ~~~100.123

`printf("%.4f", 1.23456);` ➔ 1.2346

exercice

- Quel affichage produit ce programme?

```
#include<stdio.h>
```

```
main()
```

```
{ int x = 0x10+ 010+10;
```

```
printf ("x = %x", x);
```

```
printf ("x = %o", x);
```

```
printf ("x = %d", x);
```

```
}
```

Lecture formatée de données: scanf ()

- la fonction **scanf** permet de lire des données à partir du clavier
- Syntaxe : **scanf("format", AdrVar1, AdrVar2, ...);**
 - **Format** : le format de lecture de données, est le même que pour *printf*
 - **adrVar1, adrVar2, ...** : adresses des variables auxquelles les données seront attribuées. L'adresse d'une variable est indiquée par le **nom** de la variable **précédée** du signe **&**

Exemples de scanf()

```
#include<stdio.h>

main()
{ int i , j;
  scanf("%d%d", &i, &j);
  printf("i=%d et j=%d", i, j);
}
```

ce programme permet de lire deux entiers entrés au clavier et les afficher à l'écran.

Remarque : pour lire une donnée du type **long**, il faut utiliser les spécificateurs **%ld, %li, %lu, %lo, %lx**.

Exemples de scanf()

```
#include<stdio.h>
main()
{ float x;
  double y;
  scanf("%f %lf", &x, &y);
  printf("x=%f et y=%f", x,y);
}
```

ce programme permet de lire un réel simple et un autre double à partir du clavier et de les afficher à l'écran

Remarque : pour lire une donnée du type **double**, il faut utiliser **%le** ou **%lf** et pour lire une donnée du type **long double**, il faut utiliser **%Le** ou **%Lf**