C - Affichage avec `printf`

La fonction `printf()` est la principale instruction d'affichage du langage C. Elle permet d'écrire du texte, des nombres ou des variables dans la console, souvent pour informer l'utilisateur ou vérifier le comportement d'un programme. Très flexible, elle utilise des spécificateurs de format comme %d, %f, %c ou %s pour afficher différents types de données.

5TTR 6TTR Découverte

Objectifs

À la fin de ce chapitre, tu sauras :

- Comprendre à quoi sert `printf()`.
- Afficher du texte et des variables à l'écran.
- Utiliser les principaux **spécificateurs de format** (`%d` , `%f` , `%c` , `%s`).
- Gérer les retours à la ligne et les tabulations.

Principe

`printf()` (abréviation de *print formatted*) est la **fonction principale d'affichage** en langage C. Elle permet d'écrire du texte, des nombres ou des valeurs de variables dans la **console**. Elle fait partie de la bibliothèque standard `stdio.h`, qu'il faut inclure au début du programme :

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

Exemple simple

```c
#include <stdio.h>

```

```c
int main() {
    printf("Bonjour, monde !");
    return 0;
}
```

💡 Le texte à afficher est toujours placé **entre guillemets** `" "`.

Afficher des variables

Pour afficher la valeur d'une variable, on utilise un **spécificateur de format** (`%...`) qui indique à `printf` le **type de donnée** à afficher.

Spécificateur	Type affiché	Exemple
`%d`	entier (`int`)	`printf("%d", age);`
`%f`	réel (`float`)	`printf("%f", moyenne);`
`%c`	caractère (`char`)	`printf("%c", lettre);`
`%s`	chaîne de caractères (`char[]`)	`printf("%s", nom);`

Exemple :

```c
int age = 16;
printf("J'ai %d ans.\n", age);
```

Résultat :

```
J'ai 16 ans.
```

Plusieurs valeurs dans le même texte

Tu peux afficher plusieurs variables dans une même chaîne, en respectant l'ordre des formats :

```c
int jour = 13;
char mois[] = "octobre";
int annee = 2025;

printf("Nous sommes le %d %s %d.\n", jour, mois, annee);
```

Résultat :

```
Nous sommes le 13 octobre 2025.
```

Sauts de ligne

Le caractère spécial `\n` sert à **faire un saut de ligne**. Quand tu l'utilises dans un `printf`, cela indique à l'ordinateur de **passer à la ligne suivante** après avoir affiché le texte. Sans `\n`, tout s'afficherait **sur une seule ligne** :

Exemple sans saut de ligne :

```c
printf("Bonjour");
printf("Tout le monde");
```

Résultat à l'écran :

```
BonjourTout le monde
```

Exemple avec saut de ligne:

```c
printf("Bonjour\n");
printf("Tout le monde\n");
```

Résultat à l'écran :

```
Bonjour
Tout le monde
```

Quelques caractères spéciaux

`printf()` comprend certains **caractères spéciaux** qui modifient la mise en forme du texte :

Code	Signification	Exemple
`\n`	passage à la ligne	`printf("Bonjour!\n");`
`\t`	tabulation	`printf("Nom\tAge\n");`
`\\`	affiche un antislash `\`	`printf("\\");`
`\"`	affiche une guillemet double	`printf("\"C\"");`

Formater les nombres

- Pour afficher un **nombre réel avec 2 décimales** :

```c
float moyenne = 14.357;
printf("Moyenne : %.2f\n", moyenne);
```

Résultat :

```
Moyenne : 14.36
```

Dans l'instruction `printf("Moyenne : %.2f\n", moyenne);`, la chaîne entre guillemets contient du texte et un **spécificateur de format** `%.2f` :

- `%f` indique que l'on veut afficher un **nombre réel** (de type `float` ou `double`) ;
- `.2` précise qu'on veut **afficher deux chiffres après la virgule** ;
- `\n` ajoute un **saut de ligne** à la fin du texte.

Ainsi, si la variable `moyenne` vaut `14.357`, le programme affichera :

```
Moyenne : 14.36
```

Le nombre est automatiquement **arrondi** à deux décimales pour un affichage plus lisible (selon les règles classiques des arrondis).

- Tu peux aussi aligner ou espacer les valeurs :

```c
printf("%6d\n", 42);   // Affiche "    42" (avec 4 espaces devant)
printf("%06d\n", 42);  // Affiche "000042" (complète avec des 0)
```

- `%6` indique que le nombre doit aligné à droite dans une zone de 6 caractères de long rempli par des espaces
- `%06` indique que le nombre doit aligné à droite dans une zone de 6 caractères de long rempli par des zéros
- `%d` affiche un entier

Erreurs fréquentes

❌ Oublier les guillemets autour du texte :

```c
printf(Salut);   // Erreur
```

✅ Correct :

```c
printf("Salut");
```

❌ Ne pas mettre le bon format :

```c
int x = 10;
printf("%f", x);   // Mauvais type (int ≠ float)
```

✅ Correct :

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

À retenir

- `printf()` affiche du texte ou des valeurs à l'écran.
- Le texte est entre guillemets `" "`.
- Les **spécificateurs de format** permettent d'afficher différents types de données.
- `\n` permet de **passer à la ligne suivante**.
- Il faut **faire correspondre les formats et les types** de variables.

Exemples d'erreurs courantes

Voici quelques **exemples classiques de mauvaise utilisation des modificateurs de format** dans `printf()` et leur **impact concret** sur l'affichage 👇

Mauvais format pour un entier

```
int age = 16;
printf("%f", age);   // Mauvais format
```

💬 **Problème :** `%f` sert à afficher un nombre réel (`float`), pas un entier. 🎯 **Résultat possible :** affichage d'un **nombre aléatoire ou incohérent**, car `printf` interprète les bits de l'entier comme un réel. Ou `0`. Ca dépend du compilateur. ✅ **Correct :**

```
printf("%d", age);
```

Mauvais format pour un réel

```
float moyenne = 13.5;
printf("%d", moyenne);   // Mauvais format
```

💬 **Problème :** `%d` est prévu pour un entier. 🎯 **Résultat :** la valeur affichée n'aura aucun sens (souvent un grand entier négatif). ✅ **Correct :**

```c
printf("%f", moyenne);
```

Mauvais format pour une chaîne de caractères

```c
char nom[] = "Alice";
printf("%c", nom);   // Mauvais format
```

💬 **Problème :** `%c` affiche un seul caractère, alors que `nom` est un tableau (`char[]`). 🎯 **Résultat :** l'affichage est imprévisible (souvent un symbole bizarre). ✅ **Correct :**

```c
printf("%s", nom);
```

Trop ou pas assez de variables

```c
int a = 5, b = 10;
printf("a = %d, b = %d, c = %d\n", a, b);   // Une variable manquante
```

💬 **Problème :** le dernier `%d` n'a pas de valeur associée. 🎯 **Résultat :** affichage d'un nombre aléatoire, ou plantage du programme. ✅ **Correct :**

```c
printf("a = %d, b = %d\n", a, b);
```

Ordre incorrect des formats

```c
int x = 5;
float y = 3.14;
printf("x = %f, y = %d\n", x, y);   // Mauvais ordre
```

💬 **Problème :** les variables ne correspondent pas au bon type de format. 🎯 **Résultat :** les valeurs affichées seront totalement fausses. ✅ **Correct :**

```c
printf("x = %d, y = %f\n", x, y);
```

Oublier le retour à la ligne

```c
printf("Bonjour");
printf("tout le monde");
```

💬 **Problème :** le texte s'enchaîne sans séparation. 🎯 **Résultat :**

```
Bonjourtout le monde
```

✅ **Correct :**

```c
printf("Bonjour\n");
printf("tout le monde\n");
```

À retenir

- Chaque **modificateur de format** (`%d`, `%f`, `%c`, `%s`, etc.) correspond à un **type précis**.
- Si le format ne correspond pas au type, le programme affiche des **valeurs incohérentes** (ou plante).
- Il faut toujours vérifier que **le nombre et le type** des formats correspondent exactement à ceux des variables.
