C - Les fonctions

Une **fonction** est un **bloc de code réutilisable** qui exécute une tâche précise. Plutôt que de répéter les mêmes instructions plusieurs fois, on les regroupe dans une fonction, qu'on peut ensuite **appeler** à différents endroits du programme. Les fonctions permettent d'organiser le code, de le rendre plus **lisible**, plus **court** et plus **facile à corriger**.

5TTR 6TTR Découverte

Objectifs

À la fin de ce chapitre, tu sauras :

- Définir et appeler une fonction en C.
- Comprendre la notion de **paramètres** et de **valeur de retour**.
- Utiliser la bonne **structure d'une fonction**.
- Faire la différence entre la **fonction principale** (`main`) et les fonctions que tu crées.

Structure d'une fonction

Une fonction en C suit une structure très précise :

```c
type_de_retour nom_de_fonction(paramètres) {
    // instructions
    return valeur;   // facultatif selon le type
}
```

- **type_de_retour** : indique le type de la valeur renvoyée (`int`, `float`, `char`, `void`...).
- **nom_de_fonction** : choisi par le programmeur, selon sa fonction.
- **paramètres** : variables reçues par la fonction.
- **return** : renvoie une valeur au programme principal (sauf si la fonction est de type `void`).

Une fonction simple sans paramètres

```c
#include <stdio.h>

void direBonjour() {
    printf("Bonjour à tous !\n");
}

int main() {
    direBonjour();  // appel de la fonction
    return 0;
}
```

> 💡 `void` signifie que la fonction **ne renvoie rien**.

Ici, `direBonjour()` est **appelée depuis** `main()` pour exécuter son code.

Fonction avec paramètres

```c
#include <stdio.h>

void afficherSomme(int a, int b) {
    printf("Somme = %d\n", a + b);
}

int main() {
    afficherSomme(5, 3);
    return 0;
}
```

Ici, `a` et `b` sont des **paramètres** : la fonction les reçoit, calcule leur somme et l'affiche.

Fonction qui renvoie une valeur

```c
#include <stdio.h>

int carre(int nombre) {
    return nombre * nombre;
```

```c
}

int main() {
    int resultat = carre(6);
    printf("Le carré de 6 vaut %d\n", resultat);
    return 0;
}
```

💡 Le mot-clé `return` renvoie la valeur calculée (`36` ici) au programme principal.

Déclaration avant `main`

En C, une fonction doit être **connue du compilateur** avant d'être utilisée. Si tu écris la fonction **après** `main()`, tu dois **la déclarer** avant :

```c
#include <stdio.h>

int carre(int nombre);   // ← déclaration (prototype)

int main() {
    printf("%d\n", carre(5));
    return 0;
}

int carre(int nombre) {  // ← définition
    return nombre * nombre;
}
```

Le **prototype** indique simplement au compilateur :

> "Il existe une fonction nommée `carre` qui prend un `int` et renvoie un `int` ."

Je trouve qu'il est plus lisible de placer la fonction `main` en début de programme et les autres fonctions après la fonction `main` : on sait directement où commence notre programme. Dans ce cas, il faut penser à déclarer tes fonctions avant... mais c'est un tout petit prix à payer...

Plusieurs fonctions

Tu peux évidemment écrire plusieurs fonctions dans le même programme :

```c
#include <stdio.h>

void afficherMessage() {
    printf("Programme de calcul\n");
}

int somme(int a, int b) {
    return a + b;
}

int main() {
    afficherMessage();
    printf("Résultat : %d\n", somme(4, 7));
    return 0;
}
```

À retenir

- Une **fonction** = un **nom + un rôle précis + un bloc d'instructions**.
- Les **paramètres** permettent de transmettre des valeurs.
- `return` renvoie un résultat à celui qui a appelé la fonction.
- `void` → pas de valeur renvoyée.
- Toujours **déclarer** une fonction avant de l'utiliser si elle se trouve après `main()`.
- Les fonctions rendent ton code **plus clair, plus structuré et réutilisable**.