C - Comment documenter correctement une fonction

Documenter une fonction permet de comprendre rapidement **à quoi elle sert**, quels paramètres elle reçoit et ce qu'elle renvoie. Une bonne documentation rend ton code plus lisible, plus professionnel et beaucoup plus facile à maintenir.

5TTR 6TTR Découverte

Introduction

Objectifs

À la fin de ce chapitre, tu sauras :

- Écrire une documentation courte et claire pour une fonction.
- Utiliser le format de style *javadoc* (description + tags).
- Décrire les paramètres et la valeur retournée.
- Reconnaître ce qu'il faut commenter ou non.

Format recommandé (style javadoc)

Voici le modèle que tu utiliseras pour documenter tes fonctions :

```
/**
 * Phrase courte : dit ce que fait la fonction, en une seule phrase.
 *
 * Description plus longue (facultatif) : précise le fonctionnement,
 * les cas particuliers, les effets de bord, etc.
 *
 * @param x  description du paramètre x
 * @param y  description du paramètre y
 * @return   description de la valeur retournée
 */
int exemple(int x, int y) {
```

```
    return x + y;
}
```

Fonction simple sans paramètres

```
/**
 * Affiche un message de bienvenue.
 *
 * Utilise printf pour écrire un texte simple dans la console.
 */
void afficherMessage() {
    printf("Bienvenue !\n");
}
```

Fonction avec paramètres

```
/**
 * Calcule la moyenne de deux nombres réels.
 *
 * Les deux valeurs sont additionnées, puis divisées par deux.
 *
 * @param a  premier nombre
 * @param b  deuxième nombre
 * @return   la moyenne de a et b
 */
float moyenne(float a, float b) {
    return (a + b) / 2;
}
```

Bonnes pratiques

À faire

- Toujours commencer par **une phrase courte**, terminée par un point.
- Rédiger ensuite (facultatif) une description plus longue.
- Décrire brièvement **chaque paramètre** avec `@param`.
- Décrire la valeur retournée avec `@return`.

- Rester cohérent dans tout le projet.
- Documenter les **effets de bord** (variables modifiées, tableau changé…).

❌ À éviter

- Décrire ligne par ligne ce que fait le code.
- Écrire des commentaires trop longs (ce n'est pas un roman).
- Oublier de mettre à jour la documentation après une modification.
- Documenter l'évident :

```
int x;  // variable x
```

Mauvais vs bon commentaire

❌ Mauvais

```
// calcule une somme
int f(int a, int b) {
    return a + b;
}
```

✔ Bon

```
/**
 * Calcule la somme de deux entiers.
 *
 * Retourne la valeur obtenue en additionnant a et b.
 *
 * @param a  premier entier
 * @param b  deuxième entier
 * @return   la somme de a et b
 */
int somme(int a, int b) {
    return a + b;
}
```

À retenir

- Une bonne documentation doit être **courte, utile et précise**.
- Le modèle javadoc aide à structurer ce que tu écris.
- On documente surtout **le but**, les **paramètres**, les **retours**, et les éventuels **effets de bord**.
- La qualité du commentaire reflète la qualité du code.