Fiche récap : Apprenez à programmer en C

Développement

1. Définissez par écrit ce que vous voulez faire : ce qui se conçoit bien s'énonce clairement !
2. Découpez le plus possible la logique de votre code sous-logiques.
3. Écrivez chaque bout de code l'un après l'autre en prenant soin de bien les tester à chaque fois.
4. Commentez votre code au fur et à mesure pour qu'il reste facilement maintenable et évolutif.

c_cpp

```c
#include <stdio.h>

// définition de la fonction increment de type void
void increment(int* nombre) // paramètre : pointeur de int
{
    (*nombre)++; // incrémente la valeur contenue dans le pointeur
}

// fonction main, le point d'entrée du programme
int main(int argc, const char * argv[]) {
    int compteur = 0; // assigner 0 à la variable compteur de type int
    // assigner l'adresse de la variable compteur
    int* compteurPointeur = &compteur;

    while (compteur < 10) // boucle tant que compteur est inférieur à 10
    {
        // affiche la valeur contenue dans le pointeur
        increment(compteurPointeur);
        printf("%d\n", *compteurPointeur);
        /* incrémente la valeur contenue dans le pointeur à l'aide
         de la fonction increment qui prend en argument un pointeur */
    }

    return 0; // fin du programme, on retourne la valeur 0
}
```

Définitions

Variable
Association d'un nom à une valeur. Elle est déclarée via un mot-clé et peut prendre plusieurs formes : texte, nombre, booléen, etc.

Opérateurs logiques
Caractères spéciaux qui permettent de lier plusieurs conditions ou de comparer des valeurs.

`&&` `||` `<` `>` `>=` `<=` `==` `!=` `!`

Boucle
Répétition d'un bloc de code tant que la condition spécifiée est valide.

Fonction
Ensemble d'instructions qui effectuent une tâche. Une fonction peut être appelée plusieurs fois dans un programme.

Pointeur
Variable spéciale qu'on utilise pour stocker des adresses plutôt que des valeurs.

Tableau
Bloc de mémoire où des données de même type sont rangées de manière contiguë (côte à côte).

Bonnes pratiques

- ✅ Nommer les variables et les fonctions de manière explicite avec la technique du "camel-case".
- ✅ Créer des fichiers d'en-tête (.h) pour déclarer les prototypes des fonctions.
- ✅ Définir les fonctions dans un fichier source (.c).
- ✅ Initialiser une variable au moment de la déclaration, si sa valeur est connue.

Erreurs classiques

- ❌ Faire des fonctions trop longues qui font trop de choses.
- ❌ Utiliser le mauvais type de variable.
- ❌ Créer une boucle infinie : si la condition est toujours vraie, le programme ne s'arrêtera jamais !
- ❌ Utiliser un pointeur **NULL**. Il ne faut pas l'utiliser tant qu'il est **NULL** sinon le programme crash.
- ❌ Ne pas mettre de point virgule à la fin d'une boucle "do...while".