Programmation (Langage en C)

#include <…>

Permet d’inclure des bibliothèques qui rajoute des fonctions au programme par exemple la bibliothèque stdio.h permet d’utiliser la fonction printf.

// Commentaires

Les // permettent la création d’un commentaire cela peut être utile pour faciliter la lecture d’un programme et pouvoir ainsi s’y retrouver plus facilement.

(pour réaliser un commentaire sur plusieurs lignes on peut utiliser /\* au début et \*/ à la fin du commentaire).

# Déclaration et affectation d’une variable

<var\_type> <var\_name> ;

Déclare une variable, lui donne un nom et permet d’y stocker une valeur en fonction du type de cette variable.

Quelques types de variable fréquent:

* Pour stocker un **entiers** on utilise **int**,
* Pour stocker un **réel** on utilise **double**,
* Pour stocker un **booléens** on utilise **bool**,
* Pour stocker un **caractères** on utilise **char**.

<var\_name> = <value> ;

Affecte une valeur à la variable cette valeur est stocker dans une zone mémoire (la valeur peut être le résultat d’un calcule ou une constante littérale mais attention car la valeur est influencée par le type de la variable.

# Entrées et sorties

printf("le nombre est <write\_format>", <var\_name>) ;

Cette fonction affiche des information à l’écran (le plus souvent des chaines de caractères et la/les valeur(s) d’une ou plusieurs variables donnée(s)).

Le write format est le format d’écriture de la valeur demandée il change en fonction du type de sa variable (les différent write format dans le tableau ci-dessous).

Scanf\_s("<read\_format>", &<var\_name>);

Scanf va lire le clavier de l’utilisateur ce qui va lui permettre d’enregistrer ce qu’il écrit dans une variable que ce soient des lettres, des chiffres ou une chaine de caractères.

Le read format est le format de lecture de la valeur écrite il change en fonction du type de sa variable (les différent read format dans le tableau ci-dessous).

|  |  |  |
| --- | --- | --- |
| Type les plus courants | <write\_format> | <read\_format> |
| Int | %d | %d |
| Long | %ld | %ld |
| Float | %f | %f |
| Double | %f | %lf |
| Char | %c | %c |

Getchar();

Le getchar permet de lire un seul caractère entrer au clavier et peut le stocker dans une variable.

Putchar(...);

Putchar va afficher un caractère stocké

# Les conditions

If (condition [opérateur logique condition]) {

Si la condition est respectée alors …

} else {

Si la condition n’est pas respectée alors …

}

|  |  |
| --- | --- |
| Opérateurs de comparaison | Traduction |
| == | Égal |
| != | Différent |
| < | Plus petit |
| > | Plus grand |
| <= | Plus petit ou égal |
| >= | Plus grand ou égal |
| Opérateurs  logique | Traduction |
| && | Et |
| || | Ou |
| ! | Non |

# Les boucles / répétitive

While (condition) {…}

While permet de répéter ce qui est entre les trémas jusqu’à ce que la condition soit respectée.

On utilise while quand on ne connait pas le nombre d’itération

For (initialization ; condition ; incrementation) {…}

For permet aussi de répéter ce qui est entre les trémas jusqu’à ce que la condition soit respectée.

Mais à la différence du while on l’utilise quand on connait le nombre d’itération.

Do {…} while (condition)

Do…While est utilisée pour pouvoir exécuter la condition si celle-ci est fausse dès le départ puisse que la condition se fait à la fin.

# Les Tableaux

Int nomDuTableau[nombreDeCases];

Permet la création d’un tableau contenant le nombre de cases souhaitée.

nomDuTableau[caseSouhaitée] = valeur ;

Permet d’inscrire une valeur dans n’importe qu’elle case souhaitée.

ATTENTION !

Si on crée un tableau de 3 cases celui-ci contiendras les cases de 0 à 2 et non pas de 1 à 3, la première case d’un tableau commence par 0.

# Les Fonctions

Type nomFonction(paramètres) {…}

Permet la création d’une fonction pour découper un morceau de code qui plus tard pourras être rappeler dans le programme.

Les paramètres sont les valeurs que la fonction va utiliser.

# Source utilisée

<https://openclassrooms.com/fr/courses/19980-apprenez-a-programmer-en-c>

Cours de langage en C hénallux