

Avant de commencer :

- 1) Installer le Cigwin.
- 2) Installer le Migwin.
- 3) Configurer les variables d'environnement.
- 4) Installer Eclipse pour C/C++.
- 5) Voir si tout marche bien. (Compilation surtout)

Remarques et quelques exemples:

1) entrée/sortie plus facile en C++ qu'en C

- ⇒ Les bibliothèques n'ont plus besoin de h, sauf pour des exceptions (cas des anciennes bibliothèques hérité du C)
- ⇒ **iostream** remplace : **stdio.h** et **stdlib.h**. Vous n'avez plus besoin de ces deux (sauf si vous aimez toujours les utiliser)
- ⇒ **int main(int argc, char *argv[])** est une nouvelle forme, pour récupérer des arguments de retour
- ⇒ deux instruction pour faire des sorties et entrées : **cout** et **cin**. Elles doivent être écrite comme ça :
std::cout<< sortie, std::cin>>entrée
- ⇒ pour ne pas avoir besoin d'ajouter **std::** tout le temps, on declare: **using namespace std;**
- ⇒ **cout** veut dire (console out=vers l'écran)
- ⇒ **cin** veut dire : console in (depuis l'écran)
- ⇒ **endl** veut dire end line=fin de ligne
- ⇒ les entrée/sorties de **cout/cin** peuvent être de n'importe quel type
- ⇒ pour faire plusieurs sorties : **cout<<sortie1<<sortie2**
- ⇒ pour faire plusieurs entrées : **cin>>entrée1>>entrée2**

2) Variable, types, déclaration

- ⇒ On peut déclarer les variables partout

Exemple : for (**int** i = 0 ; i < 10 ; i++)

- ⇒ Un nouveau type Booléen : **boul={true, false}**.
- ⇒ Opérateur logique or, and, xor,.. prédéfini
- ⇒ Pointeurs, plus facile :

En C :

```
int main()
{
    int *variable = NULL;
    variable = malloc(sizeof(int)); // Allocation de mémoire
    free(variable); // Libération de mémoire
    return 0;
}
```

En C++ :

```
int *variable = NULL;
variable = new int; // Allocation de mémoire
delete variable; // Libération de mémoire
```

- ⇒ **Tableau**

```
int *tableau = NULL; //déclaration d'un tableau sans allouer de mémoire
tableau = new int[20]; // Allocation de mémoire pour le tableau
delete[] tableau; // Libération de mémoire (tableau)
```

⇒ Créer de nouveaux type avec typedef

Exemple :

```
typedef struct Coordonnees Coordonnees;
// typedef permet d'éviter d'avoir à taper "struct" à chaque déclaration
struct Coordonnees
{
    int x;
    int y;
};
```

Et après :

```
Coordonnees point; // Le mot-clé struct est inutile grâce au typedef
```

Exercice 1 : essayer les codes suivants :

Exemple 1

```
#include <iostream>
using namespace std;
int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Exemple 2

```
int main()
{
    int age = 21;
    cout << "Salut, j'ai " << age << " ans" << endl;
    return 0;
}
```

Exemple 3

```
int main()
{
    int age = 21;
    char pseudo[] = "M@teo21";
    cout << "Salut, j'ai " << age << " ans et je m'appelle " << pseudo << endl;
    return 0;
}
```

Exemple 4

```
#include <iostream>
```

using namespace std;

```
int main()
{
    int age = 0;
    cout << "Quel age avez-vous ?" << endl;
    cin >> age;
    cout << "Ah ! Vous avez donc " << age << " ans !" << endl;
    return 0;
}
```

Exercice 2 : (usage de fichiers et de fonctions)

1. Créer un fichier txt « fichier.txt » dans un répertoire (à retenir le chemin physique de ce fichier). Remplis ce fichier texte.
2. On vous propose le code suivant d'une fonction qui permet de lire les informations à partir de ce fichier.

```
void maFonction()
{
    FILE* fichier = NULL; //déclarer un fichier
    fichier = fopen("E:/exemple.txt", "r");
    //ouvrir le fichier en lecture seule
    char Buffer[128]; //lecture du contenu dans des buffers de 128 char
    while (fgets(Buffer,128,fichier))
        // lecture tanque le fichier n'est pas terminé
        // la fonction fgets permet de lire une chaine de char
        du fichier
        printf("%s",Buffer); //afficher le contenu lu
    fclose(fichier); //fermer le fichier
}
```

3. Ecrire un programme principal qui utilise cette fonction et afficher le contenu de ton fichier ;
4. Créer un fichier txt « fichier2.txt » dans un répertoire (à retenir le chemin physique de ce fichier). Remplis ce fichier texte de manière à mettre dans chaque ligne trois valeurs numériques entières et séparées par un espace.
5. On vous propose le code suivant d'une fonction qui permet de lire les informations à partir de ce fichier.

```
void maFonction2(){
    FILE* fichier = NULL;
    int x, y, z; // on récupère ici les valeurs à chaque fois
    fichier = fopen("E:/exemple2.txt", "r");
    while (!feof(fichier)) // tanque on n'a pas atteint la fin du fichier "end of file"
    {
        fscanf(fichier, "%d %d %d", &x, &y, &z);
        //lire trois valeurs entiere du fichier
        cout<<"Valeurs: "<<x<<" "<<y<<" "<<z<<"\n";
        //afficher ces trois valeur sur écran
        //le parcours dans le fichier est fait automatiquement
    }
    fclose(fichier);
}
```

- }
6. Ecrire un programme principal qui utilise cette fonction et affiche le contenu de ce fichier ;

Exercice 3 : Reprendre tous les exercices de la Série du TD 1. Programmer en C++.