## Différences avec Python

### **1.1 Compilation**

Le C++ comme le C est un langage compilé. C'est-à-dire que lorsque le code est écrit, il faut le compiler avant de pouvoir l'exécuter.

Le code .cpp est traduit grâce à un exécutable en .o qui sont des fichiers exécutables.

Le préprocesseur: avant la compilation, le préprocesseur va effectuer quelques instructions (celle qui commence par un #).

En réalité les étapes sont plus complexes:

* préprocessing : Le code source original est transformé en code source brut. Les commentaires sont enlevés et les directives de compilation commençant par # sont d'abord traités pour obtenir le code source brut ;
* compilation en fichier objet : les fichiers de code source brut sont transformés en un fichier dit objet, c'est-à-dire un fichier contenant du code machine ainsi que toutes les informations nécessaires pour l'étape suivante (édition des liens). Généralement, ces fichiers portent l'extension .obj ou .o ;
* édition de liens : dans cette phase, l'éditeur de liens (linker) s'occupe d'assembler les fichiers objet en une entité exécutable et doit pour ce faire résoudre toutes les adresses non encore résolues, tant des mémoires adressées que des appels de fonction. L'entité exécutable est généralement soit un exécutable, soit une bibliothèque dynamique (DLLs sous Windows et toutes les variantes, tels que objet COM, OCX, etc, et les .so sous Linux).

### **1.2. Les fichiers d’entêtes**

Les fichiers "entêtes" ("headers" en anglais), on traditionnellement une extension en .h ou .hpp (mais la plupart des entêtes systèmes du C++ standard n'ont plus d'extension du tout). Ces fichiers contiennent généralement les prototypes de différentes fonctions, structures et classes.

Les fichiers d'entête contiennent la définition des méthodes avec leurs signatures. C'est-à-dire avec la définition des types que ces fonctions prennent en paramètre et doivent retourner.

Ces prototypes proviennent :

1. des bibliothèques standards du C++ : tout compilateur C++ doit fournir ces fichiers ainsi que les fichiers objets contenant l'implémentation des bibliothèques standards (souvent liées par défaut) ;
2. de bibliothèques non standards fournis par l'éditeur du compilateur ou de l'environnement de développement ;
3. de bibliothèques non standards (gratuites ou payantes) que le programmeur s'est procurées. Lorsqu'une bibliothèque non standard est utilisée et que celle-ci englobe de nombreuses fonctionnalités (interface graphiques, accès à une base de données, surcouche système, communications réseaux, etc.), on parle parfois de [framework](https://fr.wikipedia.org/wiki/Framework). Citons par exemple [Qt](https://fr.wikipedia.org/wiki/Qt) et [WxWidgets](https://fr.wikipedia.org/wiki/WxWidgets) dans cette catégorie.

Exemple de déclaration de fonction dans le header.

double Moyenne(double x, double y);

char LireCaractere();

void AfficherValeurs(int nombre, double valeur);

### **1.3. Autres**

C++ utilise des { } pour délimiter les blocs des boucles for et des if. l'indentation n' est plus nécessaire comme dans python mais reste une bonne pratique.

double Moyenne(double x, double y){

return x\*y;

}

//ou si la méthode est une méthode de la classe Calcul

double Calcul::Moyenne(double x, double y){

return x\*y;

}

## 2. Déclaration et affectation

Cette fois, contrairement à Python, la création et l’affectation peuvent être faites en deux étapes. Et il faut aussi préciser le type de la variable lors de la déclaration (**typage fort**).

La création ou déclaration:

int i;

std::string s;

ou plusieurs d’un coup:

int i, j, k;

L’affectation:

i=0;

str=0; par contre n’est pas possible puisque str est une string. Le programme ne pourra pas compiler.

Les deux étapes peuvent être faite en une seule fois:

int i=0;

ou plusieurs d’un coup:

int i, j = 0, k;

En particulier, on peut déclarer une ‘variable’ comme étant **constante** en ajoutant const devant le nom du type, par exemple :

const double PI = 3.14159265358979323846;

Une constante ne pourra plus être modifiée par la suite.

## 3. Types

|  |  |  |
| --- | --- | --- |
| **Type** | **Longueur** | **Domaine de valeurs** |
| unsigned char | 1 octet | 0 à 255 |
| char | 1 octet | -128 à 127 |
| enum | 2 octets | -32,768 à 32,767 |
| short (int) | 2 octets | −32 768 à 32 767 |
| int | 4 octets | −2 147 483 648 ; 2 147 483 647 |
| unsigned (int) | 4 octets | 0; 4 294 967 295 |
| unsigned long | 5 octets | 0 ; 18 446 744 073 509 552 000 |
| long | 5 octets | −9 223 372 036 854 775 808 ; 9 223 372 036 854 775 807 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type de Base** | **Taille du type en octets** | **Taille de l'exposant** | **Taille de la mantisse** | **Nombres de valeurs possibles** | **Plage de valeurs possible** |
| float | 4 | 8 bits | 23 bits | 4 294 967 296 | (indisponible actuellement) |
| double | 8 | 11 bits | 52 bits | 18 446 744 073 509 552 000 | (indisponible actuellement) |
| long double | 12 | 15 bits | 64 bits | 79 008 162 513 705 380 000 000 000 000 | (indisponible actuellement |

Le mot unsigned précédant le nom d’un type entier (on parle alors d’entier non signé), un tel entier est toujours positif.

**Attention!!**

Le type boolean (True, False en C++), est en fait 0 pour faux et le reste (généralement 1 pour vrai).

### **3. 1 Définitions de nouveaux types à partir de type existant**

Avec **typedef**

const int DIM = 3;

typedef float Vecteur[DIM]; // définit un type Vecteur (= tableau de trois réels)

typedef int Matrice[2][3]; // définit un type Matrice

typedef char Phrase[256]; // définit un type Phrase

### **3. 2 Définition d’énumération**

enum Suit { Diamonds, Hearts, Clubs, Spades };

## 4. Opération sur les entiers et flottants

|  |  |  |  |
| --- | --- | --- | --- |
| **Opérateur** | **exemple** | **raccourci** | **type** |
| + (addition) | 5+3 → 8 | var=5  var+=3 → var=8 | int+int → int  int+float → float |
| - (soustraction) | 5-3 → 2 | var=5  var-=3 → var=2 | int-int → int  int-float → float |
| \* (multiplication) | 5\*3 → 15 | var=5  var\*=3 → var=15 | int\*int → int  int\*float → float |
| / division | 6/3 → 2.0 | 19.0 / 5.0 → 3.8  19 / 5 → 3 | int/int → int  int/float → float |
| % (le reste) | 5%3 → 2  car 5= 3\*1+2 | var=5  var%=3 → var=2 | int%int → int  int%float → float |

Contrairement à Python le / entre entiers fait directement l’équivalent du //.

Il existe aussi de nouveau ‘opérateur’.

++ (incrémente la variable de 1)

-- (décrémente la valeur de 1)

Exemple:

int c =0;

c++; (maintenant c vaut 1)

**Opérateurs de comparaison (pareil que Python)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| == (égal à) | < (inférieur à) | > (supérieur à) | >= (sup ou égal à) | <= (inf ou égal à) | !=  (différent de) |

Ces opérations vont renvoyer des booléens, c’est-à-dire un résultat qui vaudra : Vrai ou Faux (*True* or *False*)

## 4. Fonctions importantes

### **4.1 Print → cout (**pour afficher du texte dans la console):

cout <<

Exemple:

std::cout << "Vous vous appelez " << total << "." << endl;

option du cout:

* endl provoque un passage à la ligne
* setfill(c) fixe le caractère de remplissage (si ce n’est pas un blanc)
* setprecision(p) fixe le nombre de chiffres affichés
* setw(n) fixe la largeur de l’affichage
* setbase(b) fixe la base de numération

### **4.2 Input → cin** (pour demander de saisir une valeur):

cin >>

Généralement on veut garder en mémoire la donné saisie (donc avec une variable) var=input()

### **4.3 Conversion de type (cast)**

int(), str(), float() pour convertir une donnée en entier, chaine de caractère ou flottant.

int(‘5’) → 53, float(“5”) → 5, mais beaucoup d’autre cast ne marche pas de cette facon.

Lorsque c’est possible, il faut utiliser le **static\_cast**<int>(5.2);

### **4.4 Et plein d’autres**

<https://docs.python.org/fr/3.6/library/functions.html>

Pour les connaitres, on peut utiliser la fonction help(input())

### **4.5 Création de fonction**

Une nouvelle fonction se déclare en mettant

type renvoyé nom De fonction

{

…

}

Exemple:

*int main() //Debut de la fonction main() et donc du programme*

*{*

*cout << "Bonjour tout le monde !" << endl;*

*return 0;*

*} //Fin de la fonction main() et donc du programme*

## **5. Listes (et Tableaux)**

Cette fois les deux types sont bien différents.

### 5.1 Tableaux

1. **Pour un tableau à une taille fixe**

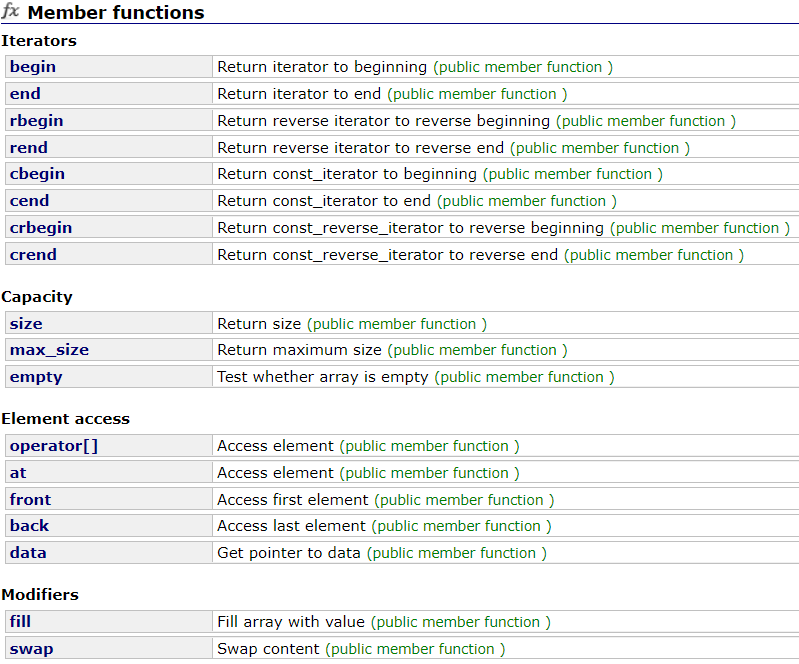
type nom[taille]

Exemple:

int meilleurScore[5]; //Déclare un tableau de 5 int

double anglesTriangle[3]; //Déclare un tableau de 3 double

Note: On peut également utiliser le type array de la bibliotheque std.



1. **Pour un tableau à plusieurs dimensions**

type nomTableau[tailleX][tailleY]

1. **Pour déclarer un tableau de taille variable**

vector<vector<int>> grille;

pour ajouter des élément dans ce type de tableau, on utilise la méthode push\_back();

Ex: v1.push\_back(10); //ajoute 10 à la fin du tableau



1. **Pour accéder à un élément**

meilleurScore[2] = 5;

### **5.2 Listes**

En fait le type liste n'est pas équivalent à celui de python qui est un peu mélange de tableau et de liste.

Mais on peut utiliser les listes doublement chaînées de la bibliothèque std::list.

Sinon, on peut également utiliser les pointeurs pour définir nos propres types de listes, de piles, ou de file.

La liste doublement chaînée de la librairie std, possède les méthode suivantes.



### **5.3 Enumération**

enum Jour {dimanche, lundi, mardi, mercredi, jeudi, vendredi, samedi};

## **6. Les boucles**

Itérer avec un nombre:

*for(int i(0); i<nombreMeilleursScores; ++i)*

*{*

*cout << meilleursScores[i] << endl;*

*}*

*int n, somme = 0;*

*int i = 0;*

*cout << "Quand on le demande, tapez un entier suivi de <entrée>\n";*

*while (i < 10) // boucle version while*

*{*

*cout << "Entr´ee ? ";*

*cin >> n;*

*somme += n;*

*i++;*

*}*

*cout << "La moyenne est " << somme / 10.0;*

***Avec un do***

*do*

*{*

*cout << "Entr´ee ? ";*

*cin >> n;*

*somme += n;*

*i++;*

*}*

*while (i < 10);*

**Avec un for :**

*for (i = 0; i < 10; i++)*

*{*

*cout << "Entrée ? ";*

*cin >> n;*

*somme += n;*

*}*

## **7. Tests**

### **7.1 If, Else if, Else**

If = Si, else if = sinon, si … else = Sinon

if (condition1) {

// block of code to be executed if condition1 is true

} else if (condition2) {

// block of code to be executed if the condition1 is false and condition2 is true

} else {

// block of code to be executed if the condition1 is false and condition2 is false

}

Les opérateurs sont différents du Python, pas de and or not mais plutôt &&, || et !

#### **Condition: && (and), || (or) et ! (not)**

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **A && B** | **A || B** |
| 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 |

Pour la négation, on utilise !: !(1) → 0, !(0) → 1

### **7.2 Switch**

On a aussi le switch:

switch (note / 2) // on suppose note entier entre 0 et 20

{

case 10 :

mention = "TB et Félicitations";

break;

case 9 :

mention = "TB et Félicitations";

break;

case 8 :

mention = "TB";

break;

case 7 :

mention = "B";

break;

case 6 :

mention = "AB";

break;

case 5 :

mention = "P";

break;

default :

mention = "AJOURNE";

}

# 8. Imports

Déclaration des imports:

#include <iostream.h>

#include "Arme.h"