Chapitre I

Notions de bases de la programmation en C++

I.1. Introduction

Le langage C++ a été conçu par Bajarne Stroustup[1] en 1983, il fait partie des langages de programmation les plus utilisés dans le monde en raison de sa rapidité d'exécution, la richesse de sa bibliothèque, la variété de ses techniques de programmation...etc. Il est entièrement basé sur le langage C mais avec plusieurs nouveautés comme la possibilité de la programmation orientée objet [1]. Dans ce chapitre, nous allons étudier les notions de base de la programmation en C++. Dans la première partie, les logiciels nécessaires pour la programmation en C++ seront présentés. Ensuite, nous présentation les règlements de la déclaration et d'utilisation des constates, les variables, et les références. Dans la dernière partie de ce chapitre, nous allons étudier les différentes structures de contrôle conditionnel et itératif en C++. Plusieurs exemples d'application et des exercices avec solutions seront présentés pour faciliter la compréhension des notions étudiées.

I.2. Logiciels nécessaires pour programmer en C++

Pour écrire et exécuter un programme C++, nous allons besoin au trois logiciels installés sur notre ordinateur :

1. Editeur de texte (en anglais : Editor)

Editeur de texte est un logiciel permet d'écrire le code source du programme. Le Bloc-Notes de Windows ou ***vi*** du Linux peuvent remplir cette tâche. Néanmoins, il est très recommandé d'utiliser un éditeur de texte intelligent qui peut colorer automatiquement les mots clés, ouvrir et fermer les parenthèses…ce qui rend le code du programme lisible et compréhensible.

2. Compilateur (en anglais : compiler)

Le compilateur est un logiciel permet de traduire le programme C++ en langage machine (binaire), ce dernier sera ensuite exécuté par le système d'exploitation de l'ordinateur.

[1] Informaticien Danois né le 30 décembre 1950.

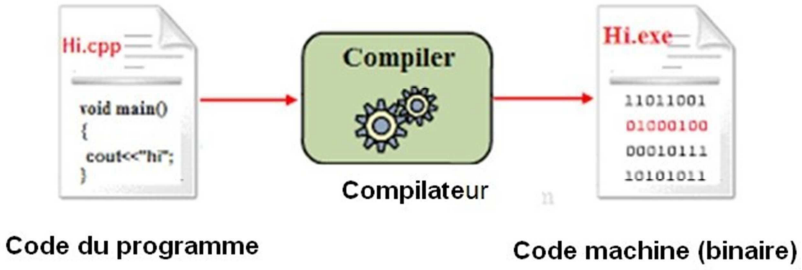

Figure (I.1). Fonction du compilateur : il permet de faire une transformation du code C++ en code machine.

3. Débogueur (en anglais : debugger)

Le débogueur est un logiciel permet de signaler au programmeur les erreurs du programme, par exemple erreur à l'écriture d'une instruction, syntaxe non respectée, manque d'une parenthèse ou point-virgule…etc.

Remarque : Dans le marché, on peut trouver les trois programmes précédents (Editeur de texte, Compilateur et Débogueur) combinés dans un seul logiciel appelé IDE (Integrated Development Environment), on citons par exemple '*Code::Blocks*' et '*Visual C++*'.

I.3. Notions de base de la programmation en C++

I.3.1. Utilisation des commentaire en C++

Les commentaires sont des phrases ou des paragraphes ajoutés au code du programme pour le but d'expliquer son fonctionnement ou pour ajouter des explications à l'attention du lecteur. Ils n'ont aucun impact sur le fonctionnement du programme (c.à.d. le compilateur ne les exécute pas). En C++, il y a deux types de commentaires :

Commentaires courts : les commentaires courts peuvent être insérés sur une seule ligne en utilisant les double barre oblique (//).

Exemple:

```
Cout <<"Bonjour les étudiants" endl ; //cette instruction affiche une phrase sur l'écran de l'ordinateur.
```

Commentaires longs : les commentaires longs peuvent s'étendre sur plusieurs lignes. Le texte doit être placé entre les symboles /* et */

Exemple:

```
Cout <<"Bonjour les étudiants" endl ;
/* L'instruction précédente affiche la chaîne de caractères (phrase) Bonjour les étudiants sur
l'écran de l'ordinateur ; */
```

I.3.2. Variables en C++, déclaration et utilisation

Une variable est une partie de mémoire dont sa contenue (nombre, adresse…) peut être modifiée ou initialisée pendant l'exécution du programme.

I.3.4.1 Types de variables

En C++, il existe plusieurs sortes de types de variables, voici quelques types de variables les plus utilisées :

bool: variable de type booléen, elle peut contenir seulement deux valeurs, true (1), ou false (0) (1bit).

char: variable de type caractère peut contenir un caractère (a, z, ..@, ...) (01octet)

int: variable de type entier, peut contenir un nombre entier dont sa valeur est comprise entre -32768 et 32767 (02 octets).

unsigned int: variable de type entier positif, peut contenir un nombre entier positif dont sa valeur est comprise entre 0 et 65535(02 octets).

double, float: variables de type réel.

String: une chaîne de caractères qui peut contenir un mot ou une phrase.

I.3.4.2 Syntaxe de la déclaration des variables en C++

La déclaration des variables consiste à définir le nom, le type, et la valeur du variable initiale de la variable comme suit :

Type nom (valeur initiale), ou Type nom=valeur initiale ;

Exemple :

```
unsigned int age (30);     // variable entière positive nommée age et contient une valeur égale 30.
int mesure = (-1500);      // variable entière nommée mesure et contient une valeur égale -1500.
int x ;       // variable entière nommée x et non initialisée.
double ksi= 0,000000312 ; // variable réelle nommée ksi et contient une valeur de 0,000000312
char lettre ('z'); // variable char (en anglais: character), nommée lettre et contient le caractère z.
string spécialité ("Electronique") ; /*variable string nommée spécialité et contient la chaîne de caractères : Electronique */
```

Remarque :

- Pour la déclaration des variables de type **caractère**, il faut mettre la valeur initiale entre deux guillemets (' ') ;
- Pour la déclaration des variables de type **string**, il faut mettre la valeur initiale entre guillemet doublé (" ");

I.3.3. Références en C++, déclaration et utilisation

La référence est une étiquette accrochée à une case mémoire (variable par exemple). La notion de référence est propre au langage C++, elle n'existe pas en langage C ou Pascal.

Syntaxe de la déclaration : **type& nom** (nom de la variable à accrocher).

Où le type de référence doit être le même de celui de variable à accrocher, par exemple une référence de type 'int' ne peut pas accrocher avec une variable de type double ou float.

Exemple:

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7      int age (22);// Déclaration d'une variable entière intialisée par 22
8      int& ref_age (age);// Déclaration d'une référence accrocher au variable:age
9
10     cout <<"Votre age est:"<< age << endl;// Affichage via la variable: age
11     cout <<"votre age est:" << PI<< endl;//Affichage via la référence:ref_age
12 return 0;
13 }
```

Dans le programme ci-dessous, on a déclaré une variable 'age' de type entier initialisée par 22 (ligne 7). Dans la ligne 9, on a déclaré une référence 'ref_age' accrochée au variable 'age'. Dans la ligne 11, on a affiché le contenu de la variable 'age' par la méthode habituellement utilisée. Par contre, dans la ligne 13, on a affiché le contenu de la variable 'age' en utilisant cette fois ci la référence 'ref_age'. **Donc on a pu accéder au contenue de la variable 'age' via la référence 'ref_age'.**

I.3.4. Constantes, déclaration et utilisation

Une constante est une partie de mémoire dont son contenu ne peut pas être changé durant l'exécution du programme. Les constantes sont déclarées en écrivant le mot-clé '**const**' suivi par le type, le nom et la valeur de chaque constante.

Exemple:

```
const float PI (3.1415) ; //Déclaration d'une constante nommée PI de type float et de valeur 3.1415
const char BEEP ('\b') ; // Déclaration d'une constante nommée BEEP de type char, et contient //le caractère \b (bip sonore).
```

I.3.5. Opérateurs standards en C++

I.3.5.1. Opérateurs arithmétiques

Les opérations arithmétiques sont utilisées pour les calculs numériques. Il y a cinq opérateurs arithmétiques sont : + (Addition), - (Soustraction), ***** (Multiplication), / (Division), **%** (Modulo ou reste d'une division, par exemple 11%3 donne 2, 15%3 donne 0).

I.3.5.2. Opérateurs de comparaison

Les opérateurs logiques sont utilisés généralement pour faire une condition sur l'exécution ou non d'une ou multiple instruction.

&& (et logique), *exemple*, 1 && 0 donne 0, 1 && 1 donne 1.

'| |' (ou logique), *exemple*, 1| | 0 donne 1.

'!' (négation logique), *exemple* !1 donne 0, !2018 donne 0.

== (est-il égal à ?, *exemple* x==10, y+1== 04.

!= (différent de ?), *exemple* x !=10.

<, <=, >,>= (inférieur de, inférieur ou égal de, supérieur de….).

I.3.5.3. Opérateurs d'affectation composée

+= (ajouter à)

-= (diminuer de)

*= (multiplier par)

/= (diviser par)

%= (modulo)

Exemple:

```
#include <iostream>
#include <string>  // Bibliothèque string
using namespace std;
int main()
{
    int a(4),b(2);// Déclaration de deux variables (a et b)de type
                  // entier initialisée respectivement à 4 et 2

    a+=2;   // a vaut 6
    b-=2;   // b vaut 0
    a *=3;  // a vaut 18
    a /=3;  // a vaut 6
    a %=5;  // a vaut 1
    return 0;
}
```

I.3.5.4. Opérateurs d'incrémentation et de décrémentation

Evidemment, pour incrémenter une variable x, on utilise x=x+1, et pour le décrémenter, on utilise x=x-1. Cependant, on peut effectuer ces opérations de manière plus simple et plus avantageuse grâce aux opérateurs ++ (incrémentation par 1) et -- (décrémentation par 1).

Exemple : le programme montre l'utilisation des opérateurs d'incrémentation et de décrémentation

```
#include <iostream>
using namespace std;
int main()
{
    int x(10),y(5);// Déclaration de deux variables(x et y) de type
                   // entier initialisée respectivement à 10 et 5
    x++;// x vaut 11
    y--;// y vaut 4
    ++x;// x vaut 12
    ++y;// y vaut 5
    return 0;
}
```

Remarque importante

Si on a une incrémentation/décrémentation et une affectation en même temps, **il y une différence** importante entre les écritures (x++, x--) et (++x, --x). La différence entre les deux écritures est illustrée dans l'exemple suivant:

z = x++; // **Passe d'abord** la valeur de x a z, puis incrémenter.
z = ++x ; // Incrémenter d'abord la valeur de x et **passe après** la valeur incrémentée à z.
Supposant que les valeurs initiales de z et x étaient respectivement 0 et 1, après l'exécution de deux instructions, elles deviennent :

z=0, x=1;
z = x++; // z vaut 2 et x vaut 1 (**Passe d'abord** la valeur de x a z, puis incrémenter).
z = ++x; // z vaut 2 et x vaut 2 (Incrémenter d'abord x et **passe après** la valeur incrémentée à z).

I.3.5.5. Opérateur d'entrée et de sortie

Pour afficher une valeur numérique ou une chaine de caractères sur l'écran de l'ordinateur, on utilise le mot clé '**cout**' suivi par l'opérateur de sortie '**>>**' et la valeur à afficher (nombre, mot, phrase,...).

Exemple:

```
cout << 2018; // Affichage des valeurs numériques
cout <<"Bonjour les étudiants"; // Affichage d'une chaine de caractères (mot ou phrase).
```

Pour lire au clavier une valeur numérique ou une chaine de caractères, on utilise le mot clé '**cin**' suivi par l'opérateur d'entrée '**>>**' et le nom de la variable à utiliser pour stocker la valeur entrée.

Exemple:

```
int age(0) ; string nom ("0") ; // Déclaration de deux variables
cout << "Quel est votre nom ?" <<endl; //Demande de l'utilisateur d'introduire son nom
cin >> nom ; // lecture de la chaine de caractères introduite et la stocker dans la variable 'nom'.
cout<< "Quel est votre age ?"<<endl; //Demande de l'utilisateur d'introduire son age
cin>> age; // lecture du nombre introduit par l'utilisateur et le stocker dans la variable 'nom'.
```

Exercice :

Ecriviez un programme qui demande à l'utilisateur d'introduire son nom, son prénom, son âge et sa spécialité, puis afficher toutes ces informations dans une seule phrase.

Solution : voir la page suivante