```cpp
#include <iostream>
#include <string>  // Bibliothèque string
using namespace std;
int main()
{
    string nom("0"),prenom("0"),specialite("0");// Trois variables de type
                                                // string
    int age (0); // une variable de type entier
    cout<< "Quel est votre nom ?"<<endl;//Demander l'âge de l'utilisateur
    cin>> nom ; // lire et stocker le nom de l'utilisateur
    cout<< "Quel est votre prenom ?"<<endl;//Demander le prénom d'utilisateur
    cin>> prenom;// lire et stocker le pénonom de l'utilisateur
    cout<< "Quel est votre specialite ?"<<endl;
    cin>> specialite;
    cout<< "Quel est votre age ?"<<endl;//
    cin>> age;
    /* l'instruction c-dessous affiche le nom le prénom, l'âge et la spécialité
    de l'utilisateur*/
    cout<< "Bonjour "<< nom <<" "<<prenom <<" vous avez "<<" "<<age<<
     " ans" <<" et votre specialite est " <<specialite <<endl;
    return 0;
}
```

I.3.6. Structures de contrôles alternatives

Les structures de contrôles alternatives permettent de définir la suite dans laquelle les instructions sont exécutées, c.à.d. elles permettent d'imposer des conditions pour l'exécution ou non d'un bloc d'instructions.

I.3.7.1. Structure de contrôle alternative if-else

La syntaxe d'utilisation de la structure de contrôle if-else est la suivante :

```
if (Conditions)
    <bloc d'instructions 1>
else
    <bloc d'instructions 2>
```

Si les conditions (une ou multiples) sont vérifiées, alors le *<bloc d'instructions 1>* est exécuté et le *<bloc d'instructions 2>* est ignoré.

Si les conditions ne sont pas vérifiées, alors le *<bloc d'instructions 2>* est exécuté et le *<bloc d'instructions 1>* est ignoré.

Remarque : le <bloc d'instructions> peut-être :

- Multiple d'instructions placées impérativement entre deux accolades ;
- une seule instruction placée facultativement entre deux accolades.

Exemple : Le programme suivant lit deux nombres différents (a, b) entrés au clavier et affiche le plus grand des deux nombres.

```cpp
#include <iostream>
using namespace std;

int main()
{
    double a(0), b(0);// Déclaration des variables a et b

    cout << "Introduisez trois nombres différent:" << endl;
    cin >> a >> b; // lire det stocker les deux nombres
                   // dans les varaibels a,b

    if (a>b) // si la condition 'a>b' est vérifée l'instruction suivante
             // sera exécutée
    {
     cout << " le premier nombre " << a << " est le plus grand";
    }
    else // si la condition 'a>b' n'est pas vérifée l'instruction suivante
         // sera exécutée
    {
     cout << " le deuxième nombre " << b << " est le plus grand";
    }
return 0;
}
```

I.3.7.2. Structure de contrôle alternative if-else if- …-else

Dans le cas de plusieurs alternatives (conditions), il est possible de combiner plusieurs structures if-else, la syntaxe dans ce cas est comme suit :

```
if (Conditions 1)
    <bloc d'instructions 1>
  else if (Conditions 2)
    < bloc d'instructions 2>
        .
        .
  else if (<Conditions n>)
    < bloc d'instructions n>
  else     // Facultative, elle est exécutée lorsqu'aucune condition n'a été vérifiée
    < bloc d'instructions n+1>
```

Les conditions < Conditions 1 >, < Conditions 2 >... < Conditions n > sont évaluées l'une après l'autre jusqu'à ce que l'une de celles soit vérifiée, le bloc d'instructions lié à la condition vérifiée est exécuté et le traitement de la commande est terminé. Le <bloc d'instructions n+1> sera exécuté si et seulement si toutes les conditions ne sont pas vérifiées.

Solution :

Exemple : Le programme suivant demande à l'étudiant d'introduire sa note d'examen en module 'Programmation orientée objet' puis affiche la mention correspondante à la note introduite comme suit selon la grille suivante:

Si 18≤ note <20, la mention est : Excellent ;

Si 16≤ note <18, la mention est : Très bien ;

Si 14≤ note <16, la mention est : Bien ;

Si 10≤ note <14, la mention est : Passable ;

Si 0≤ note <10, la mention est : Insuffisant ;

Si note <0, ou note >0, Demander de l'étudiant d'introduire à nouveau sa note ;

```cpp
#include <iostream>
using namespace std;
int main()
{
 double note (0);
 cout << "Introduisez votre note d'examen en module Programmation orientée objet" << endl;
 cin >> note;
 if (note <20 && note >=18)
    {
    cout << "Excellent" << endl;
    }
    else if (note < 18 && note >= 16)
    {
    cout << "Très bien" << endl;
    }
    else if (note < 16 && note >=14)
    {
    cout << "bien" << endl;
    }
    else if (note < 14 && note >=10)
    {
    cout << "passable" << endl;
    }
    else if (note < 10 && note >=0)
    {
    cout << "insuffisant" << endl;
    }
    else
    {
    cout << "Votre note doit etre entre 0 et 20" << endl;
    }
return 0;
}
```

Exercice :

Ecrivez un programme qui calcule les solutions réelles d'une équation du second degré $ax^2 + bx + c = 0$, tel que ($a \neq 0, b \neq 0, c \neq 0$).

Solution :

```cpp
#include <iostream>
#include<cmath> // pour qu'on puissent utilser les fonctions 'pow' et sqrt'
using namespace std;
int main()
{
    double a(0),b(0),c(0),delta(0);
    cout << "Introduisez les valeurs de a, b et c" << endl;
    cin >>a >> b >> c;//lecture des valeur a,b et c
    delta= pow(b,2)-4*a*c;//Calcul de delta
    if (delta>0)
    {   // delta >0==> il y a deux solutions
        double x1= (-b-sqrt(delta))/2; // première raçine
        double x2= (-b+sqrt(delta))/2; // deuxième raçine
        cout << " Il y a deux solutions: " << x1 <<"\t"<< x2<<endl;
    }
    else if(delta==0)
    {   // il y a une solution double
        double x1= (-b-sqrt(delta))/2;
        cout << "Il y a une solution: " << x1<<endl;
    }
    else
    {   // pas de solution dans l'ensemble R
        cout << "il n'y a pas de solutions dans |R: "<<endl;
    }
    return 0;
}
```

I.3.7.3. Structure de contrôle alternative « Switch »

La structure 'switch' permet de tester l'égalité du contenu d'une variable (de type int ou char) pour plusieurs valeurs constantes. En plus, elle offre une écriture alternative plus organisée et plus simple que celle de l'instruction 'if'. Néanmoins, cette structure présente les limitations suivante:

- Elle ne permet de tester que l'égalité (on ne peut pas tester la supériorité ou l'infériorité !);
- Elle ne peut être utilisée qu'avec des nombres entiers (type int) ou des caractères (type char), on ne peut pas l'utiliser avec des nombres de type double ou float par exemple;

Syntaxe de la structure switch :

```
switch (expression)
{
case constante_1 : // si la valeur de l'expression ==constante_1
    < bloc d'instructions1>
    break ; // retour (fin de test)
     . . .
case constante_n : // si la valeur de l'expression ==constante_n
    < bloc d'instructions n+1>
    break ; // retour (fin de test)
default : // si la valeur de l'expression est différente de toutes les constantes
    < bloc d'instructions n+1>
}
```

La syntaxe consiste à évaluer l'expression (placée devant switch) puis chercher la valeur de l'expression dans la liste des constantes placées devant chaque 'case'. Si la valeur est trouvée, le bloc d'instructions correspondant est exécuté et le processus est terminé. Sinon, le programme bascule vers le default (qu'est facultatif) et exécute le bloc d'instrucion n+1.

Exercice : En utilisant la structure 'switch', écrire un programme qui demande de l'étudiant d'introduire son rang de classement au Master (A, B, C, D, ou E) et affiche ensuite le coefficient qui va multiplier par sa moyenne générale, on donne :

rang= A, le coefficient est 1 ;

rang= B, le coefficient est 0.8 ;

rang= C, le coefficient est 0.7 ;

rang= D, le coefficient est 0.6 ;

rang= E, le coefficient est 0.5 ;

Solution : voir page suivante

```
#include <iostream>
using namespace std;
int main()
{
    char rang('0');
    cout << "Vous etes dans quel rang (A,B,C, D Ou E)?" << endl;
    cin >>rang;
    switch(rang)
    {
    case 'A':
        cout << "Votre moyenne sera multipliée par: "<< 1 << endl;
        break;
    case 'B':
        cout << "Votre moyenne sera multipliée par: "<< 0.8 << endl;
        break;
    case 'C':
        cout << "Votre moyenne sera multipliée par: "<< 0.7 << endl;
        break;
    case 'D':
        cout << "Votre moyenne sera multipliée par: "<< 0.6 << endl;
        break;
    case 'E':
        cout << "Votre moyenne sera multipliée par: "<< 0.5 << endl;
        break;
    default:
        cout << "Les rangs possibles sont: A,B,C,D ou E "<< endl;
    }
    return 0;}
```

I.3.7. Structures de contrôles répétitifs (les boucles)

Les boucles permettent de répéter l'exécution d'un bloc d'instructions plusieurs fois. Le nombre de répétition soit dépend d'une validation d'une condition logique ou défini par le programmeur. En C++, il existe trois types de boucles qui sont : boucle 'while', la boucle 'do-while' et la boucle 'For'.

I.3.7.1. Boucle while

La syntaxe de la boucle while est la suivant :

```
while (condition)
{
  <bloc d'instructions>
}
```

Tant que la <*condition*> est vérifiée, le <*bloc d'instructions*> est exécuté. Sinon l'exécution continue avec l'instruction qui suit le <*bloc d'instructions*>.

Exemple : *le programme suivant permet d'afficher les nombre entier de 0 à 9* .

```cpp
#include <iostream>

using namespace std;

int main()
{
    int i(0);
    while(i<10)
    {
    cout << i << endl;
    i++;
    }

    return 0;
}
```

I.3.7.2. Boucle do-while

La boucle 'do-while' est similaire à la boucle 'while' sauf que la boucle 'do-while' exécute le <*bloc d'instructions*> et évalue après la condition. La syntaxe de cette boucle est :

```
do
{
  <bloc d'instructions>
}
while (condition) ;
```

Remarque :
Avec la boucle do-while, le <*bloc d'instructions*> est exécuté au moins une fois.

Exemple: le programme suivant redemande de l'utilisateur d'introduire son âge tant que celui-ci est inférieur à 0, égal à 0 ou supérieur à 110.

```cpp
#include <iostream>

using namespace std;

int main()
{
    int age(0);
    do
    {
    cout << "Introduisez votre age " << endl;
    cin >> age;
    }
    while (age <=0 || (age >110));
    cout << "Merci d'avoir entré votre age " << endl;
    return 0;
}
```

I.3.7.3. Boucle for

La syntaxe de cette la boucle 'for' est la suivante :

```
for (initialisation; condition; mise à jour)
{
  <bloc d'instructions>
}
```

Où :

- L'initialisation permet de déclarer et/ou initialiser la (les) variable(s) de contrôle de la boucle ;
- La condition permet de décider si la boucle est répétée ou non ;

- La mise à jour permet d'actualiser la (les) variable (s) de contrôle de la boucle.

Exemple : le programme suivant permet de calculer la somme de N premiers nombres entiers (s=1+2+……N).

```
#include <iostream>
using namespace std;

int main ()
{
    int N(0);double somme(0);
    cout << "Entrer un entier positif"<<endl;
    cin >> N;
    for(int i=1;i<=N;i++)
    {
        somme+=i;
    }
cout << "la somme de N premiers entirs égale: "<<somme<< endl;
return 0;
}
```

I.4. Conclusion

Dans ce chapitre, nous avons étudié les notions de base de la programmation en C++. En premier temps, nous avons vu comment déclarer et utiliser les variables, les constantes, et les références. Nous avons vu également comment utiliser les opérateurs d'affectation simple et composé, les opérateurs d'incrémentation et de décrémentation, les opérateurs d'entrée et de sortie et les opérateurs de comparaison. Ensuite, nous avons étudié en détails les structures de contrôles alternatives conditionnelles (if, if-else, switch) et les structures de contrôles réplétives (while, do-while et for). Plusieurs exemples et des exercices avec des solutions sont présentés pour faciliter la compréhension et la maitrise des notions étudiées. Dans le prochain chapitre nous allons étudier les fonctions, les tableaux et les pointeurs.