Programmation orientée objet

Version 1.0

enseignants dpt info iuto

janv. 10, 2020

Les sujets de Cours

Contents :

CHAPITRE 1

COURS1

1.1 PROGRAMMATION ORIENTEE OBJET : C++

1.1.1 Introduction :

Le langage C++ est un langage de programmation orienté objets. **Bjarne Stroustrup** a développé ce langage dans les années 1980. Il s'agit d'une amélioration du langage C, dans lequel il a introduit les concepts d'"*objets* et de *généricité*.

- Pour être exécuté, un programme C++ doit d'abord être *compilé* en langage de bas niveau : **g++ -c Premier.cpp**. Cette étape construit un fichier **Premier.o**.
- Le programme compilé est ensuite *exécutable* si on lui fait subir une édition des liens : **g++ -o Premier Premier.o**.
- L'exécution se fait par appel à **Premier** : **./Premier**.
- Une version simplifiée de l'exécution de Premier.cpp peut se faire par : **g++ -o Premier Premier.cpp**

En C++, un programme doit au minimum débuter par les lignes suivantes et contenir une fonction **main** :

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

int main() {
   // programme principal
   int x = 5 , y = -6; // déclarations de variables
   int aux = x;
   x = y;
   y = aux;
   cout<<" x = "<<x<<" y = "<<y;
   // affichage
   return 0;
}
```

L'inclusion (*#include*) permet de gérer les entrées - sorties.

- En C++, les variables ont un type et doivent être déclarées (pas nécessairement définies).
- Chaque instruction se terminent par un **;** et les blocs d'instructions sont définis à l'aide de **{** et **}**.
- On retrouve en C++ les instructions d'affectation, les conditionnelles et les boucles.

1.1.2 Premiers programmes en C++

Le programme, défini dans **PremierProgramme.cpp** ne contient que des affichages **cout<<** et des saisies **cin>>**.

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

int main() {
 cout << " Premier Programme " << endl ; // affichage
 cout << " Entrer deux entiers " ;
 int x, y ; // déclaration de variables sans initialisation
 cin >> x >> y ; // lecture des deux valeurs des variables
 cout << " Produit " <<x * y << " Somme " <<x + y << endl ;
 // affichages des résultats
 return 0;
}
```

1. g++ -c PremierProgramme.cpp (compilation -> PremierProgramme.o)
2. g++ -o PremierProgramme PremierProgramme.o (édition des liens -> PremierProgramme).

ou simplement

g++ -o PremierProgramme PremierProgramme.cpp

ou encore

g++ PremierProgramme.cpp (a.out est l'exécutable).

Pour exécuter le programme : **./nomDeLExecutable** (./PremierProgramme ou ./a.out)

Utilisation d'une fonction simple :

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

int somme(int n) {
 // somme des n premiers entiers : 1 + 2 + ... + n
 int som = 0;
 for(int i = 1; i < n + 1 ; ++i)
   som += i;
 return som;
}

int main() {
 cout<<"somme des 12 premiers entiers : "<<somme(12)<<endl;
 // somme des 12 premiers entiers : 78
}
```

— Vous noterez qu'en C++, à la différence de java (intrégralement objet), il n'est pas nécessaire de définir une classe.
— Les types C++ sont très proches de ceux de java : int, float, double, bool, char, string, ...
— Les fonctions (simples) sont définies avec une valeur de retour typée et 0, un ou plusieurs paramètre(s) typé(s).
— Utilisation de la boucle **for**.

1.1.3 Programme un peu plus complexe :

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

int nbChiffre(int nombre) {
  // nombre de chiffres d'un nombre
  int nbChif;
  if (nombre == 0)
    nbChif = 1;
  else {
    nbChif = 0;
    while(nombre != 0) {
      nombre = nombre / 10;
      nbChif += 1; // ou encore ++nbChif;
    }
  }
  return nbChif;
}

int main() {
  int val;
  cout<<" saisir une valeur entière ";
  cin>> val;  // 543298
  cout<<" nombre de chiffres = "<< nbChiffre(val) <<endl; // 6
  return 0;
}
```

Utilisation de la boucle **while**, de l'écriture plus rapide **+=** et de la division **/**. En C++, la division entre deux entiers retourne un entier et la division de deux réels (float ou double) retourne un réel.

1.1.4 Les tableaux en C++ :

Les tableaux en C++ s'apparentent aux listes python, mais ils *ne peuvent pas s'agrandir ni diminuer en taille*. Il existe plusieurs façons de déclarer un tableau, on vous en présente deux :

1. int tab[10] ; déclare un tableaux de dix entiers (au maximum) non initialisé.
2. float tabF[5] = {3, 8.5, 5.5, -4, 12.3} ;

Les tableaux de taille fixe sont appelés **tableaux statiques** :

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

#define MAX 10

int main() {
  int tab[MAX] = { ..... };
  // il est possible de mettre moins de 10 éléments
  // ceux qui ne sont pas initialisés valent 0.
  ...
  return 0;
}
```

— Si une fonction utilise un tableau en paramètre, on ne donne pas sa taille dans la déclaration, en revanche, il faut ajouter un paramètre qui fournit la taille du tableau. Il n'existe pas de fonction qui permette d'observer la taille d'un tableau statique (comme len(l) en python).

— Une fonction ne peut pas retourner un tableau.
— On désire calculer la somme des entiers positifs d'un tableau de réels :

```
#include<iostream>
using namespace std;
# define MAX 10

float sommePositifs(float t[], int taille) {
  // paramètres t (tableau) taille (sa taille)
  float som = 0.0;
  for(int i = 0; i < taille; ++i) // assez proche du range python
    if (t[i] > 0)
      som += t[i];
  return som;
}

int main() {
  float tab[MAX] = {3, -5.8, 5.5, 9.3, -6.3};
  cout<<" somme des positifs du tableau "<<sommePositifs(tab, MAX)<<endl;
  for(int i = 0; i< MAX; ++i)
    cout<<tab[i]<<" ";
  cout<<endl;
}
```

Les affichages sont :

```
somme des positifs du tableau 17.8
3 -5.8 5.5 9.3 -6.3 0 0 0 0 0
```

Vous noterez que les 5 éléments non définis dans tab valent 0.

Comment « remplir » un tableau C++ ?

Comme il n'est pas posible de retourner un tableau comme résultat de fonction, celle que nous allons définir retournera le type **void**. De même la méthode d'affichage d'un tableau (qui ne s'affiche pas directement avec un *cout*<<) retournera également un type void.

```
#include<iostream>
using namespace std;
# define MAX 10

void lecture(float tab[], int taille) {
  // lecture des éléments d'un tableau
  cout<<" saisie des éléments d'un tableau de taille "<<taille <<endl;
  for(int i = 0; i < taille; ++i) {
    cout<<"tab["<<i+1<<"] ";
    cin>>tab[i];
  }
}
```

```
void affiche(float tab[], int taille) {
  // affichage des éléments d'un tableau
  for(int i = 0; i < taille; ++i)
    cout<<tab[i]<<" ";
  cout<<endl;
}

int main() {
  lecture(tab, 5);
  affiche(tab, 5);
}
```

Montrer l'exécution du code

CHAPITRE 2

TD TP C++

2.1 Semaine 1

2.2 TD1

2.2.1 1. Moyenne :

— Ecrivez un programme permettant de calculer la moyenne de N (saisie clavier) notes.
— Modifiez ce programme pour que N soit positif et que les notes saisies soient bien comprises entre 0 et 20.

2.2.2 2. Inverser :

— Définir une fonction de profil **int inverser(int)** permettant d'inverser les chiffres d'un nombre : par exemple *inverser(2345)* vaut *5432*.
— Définissez un programme principal testant cette fonction.

2.2.3 3. Somme en base 3 :

— Définir une fonction **int sommeBase3(int, int)** - puis un programme de test - qui étant donnés deux entiers écrits en base3 calcule leur somme en base 3. Par exemple : *2210 + 112 = 10022*

2.2.4 4. Nombres premiers et nombres parfaits :

— Écrire une fonction testant si un entier est premier ou non.
— Écrire une fonction qui détermine si un nombre est parfait ou non (la somme de ses diviseurs stricts est égale au nombre)
— Définissez un programme principal testant si un nombre est premier et fournissant tous les nombres parfaits inférieurs à 100.

2.2.5 5. Minimum des éléments d'un tableau et fréquence d'un entier :

— Définir la fonction **int minimum(int tab[], int taille)** qui retournez le minimum des éléments d'un tableau.
— Définir une fonction calculant le nombre d'occurrences d'un entier dans un tableau d'entiers.

2.2.6 6. Appartenance à un tableau :

— Écrire une fonction retournant un booléen indiquant si un entier appartient ou non à un tableau d'entiers.
— Modifiez cette fonction pour qu'elle retourne -1, si l'élément n'appartient pas et l'indice de celui-ci sinon.
— Supposez que le tableau passé en paramètre soit trié par ordre croissant, utilisez l'algorithme dichotomique pour retourner -1 ou l'indice.

CHAPITRE 3

Indices and tables

— genindex
— modindex
— search