TD1 : DEMARRER LA PROGRAMMATION EN C++

A. Objectifs du TD

- Apprentissage par l'exemple du langage C++
- Apprentissage d'un environnement de développement C++ (Qt avec Qt Creator disponible en téléchargement à l'adresse : https://www.qt.io/download-open-source/)

B. Principe du TD

- Apprentissage du langage C++ à partir d'exemples déjà programmés.
- Ce TD comporte des exercices de version et de thème alternés, ainsi que des explications de points particuliers.
- Les élèves réalisent les exercices proposés et posent des questions à l'enseignant en cas de problème.
- Chaque exemple de programme proposé doit être d'abord totalement compris et les élèves doivent pouvoir répondre aux questions avant de passer à la programmation demandée d'un exercice similaire.
- Ce TD suppose la disponibilité d'un environnement de développement C++ .
- Au départ, l'enseignant présente rapidement l'environnement de développement si nécessaire (lancement, arrêt, compilation, édition de liens, exécution, gestion des catalogues, des sources, des binaires, exemples d'erreur à la compilation, au link, à l'exécution, aide en ligne).

C. Travail à effectuer

- Comprendre les exemples proposés,
- Répondre aux questions posées,
- Programmer les exercices proposés.

D. Apprentissage de C++ par l'exemple

I. SEQUENCE 1

Notions à maitriser

- Structure d'un programme C++
- Variables
- Instruction de lecture et écriture
- Calculs élémentaires
- Instruction de choix (if(condition) instruction1 else instruction2)

Exemple 1.1 : Programme minimal de calcul de la moyenne entière de deux nombres entiers

```
/*
     ******************************************************
     *  Exemple 1.1 : Calcul de la moyenne de deux entiers *
     ******************************************************
*/
#include <iostream>    // declarations des fonctions d'E/S de bibliotheque
using namespace std;   // utilisations de noms courts pour la biliotheque
int main()             // fonction principale (main)
{
     int a,b,c;        // variables entières locales a la fonction main
     cout<<"Entrez les deux nombres entiers"<<endl; // ecriture
     cout<<"a : ";cin>>a;                    // ecriture puis lecture
     cout<<"b : ";cin>>b;                    // ecriture puis lecture
     c=(a+b)/2;                              // calcul
     cout<<"c = "<<c<< endl;                 // ecriture resultat
     return 0;
}
```

Points à noter

- Structure générale du programme
- Commentaires (/* ….. */, //)
- Directives concernant la bibliothèque (#include . . . , using namespace . . .)
- Fonction principale (int main() { . . . . . })
- Définitions de variables entières
- Ecriture et lecture, enchaînement d'écritures
- Calcul
- Renvoi d'une valeur par la fonction main()

Exemple 1.2 : Programme de résolution d'une équation du second degré

```
/*
     ************************************************************
     *  Exemple 1.2 : Resolution d'une equation du second degre  *
     ************************************************************
*/
#include <iostream>          // declarations des fonctions d'E/S
#include <cmath>             // declarations des fonctions math
using namespace std;
int main()                   // fonction principale (main)
{
     int a,b,c;           // variables entières locales a la fonction main
     double delta,x1,x2;  // variables réelles locales à la fonction main
     cout<<"Entrez les coefficients"<<endl;
     cout<<"a : ";cin>>a ;
     cout<<"b : ";cin>>b;
     cout<<"c : ";cin>>c;
     delta=b*b-4*a*c;
     if(delta < 0)
     cout<< "pas de racines reelles"<<endl;
     else
        if(delta>0)
        {
```

```
                x1=(-b+sqrt(delta))/(2*a);
                x2=(-b-sqrt(delta))/(2*a);
                cout<<"x1 = "<<x1<<endl;
                cout<<"x2 = "<<x2<<endl;
        }
        else
        {
           x1=x2=-b/(2*a);
           cout<<"racine double x1 = x2 = "<<x1<<endl;
        }
    return 0;
}
```

Points à voir

- Utilisation de la bibliothèque mathématique (#include <cmath>)
- Définition de variables réelles (double delta . . .)
- Instruction de choix (if(condition) instruction1 else instruction2)
- Opérateurs de comparaison
- Appel de la fonction de bibliothèque sqrt() (calcul d'une racine carrée)

Questions à se poser

- Pourquoi faut-il des parenthèses après main alors qu'il n'y a pas d'arguments ?
- Pourquoi y a-t-il des {} dans l'instruction if ?
- Pourquoi utilise-t-on les directives #include <iostream>, #include <cmath> ?
- Que faudrait il changer dans le programme si l'on enlevait using namespace std; ?

Programme à réaliser

Ecrire un programme qui lit un nombre entier, détermine s'il est pair ou impair et écrit sur l'écran pair ou impair.

Indication **:** l'opérateur **%** donne le reste de la division entière d'un entier par un autre, par exemple, l'expression 7%2 vaut 1.`

II. SEQUENCE 2

Notions à maitriser

- Boucles de traitement
- Mise en page des écritures

Exemple 2.1 : Programme de conversion fahrenheit-celsius

```
/*
     ************************************************************
     *   Exemple 2.1 : Table de conversion fahrenheit --> celsius *
     *   de 0 a 100 degres fahrenheit par pas de 20 degres        *
     ************************************************************
*/
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
     int inf=0,sup=100,pas=20;
     double fahr,celsius;
     cout<<"Table de conversion fahrenheit -> celsius"<<endl;
     cout<<"----------------------------------------"<<endl<<endl;
     // ----  1ere methode : boucle while     ----
     fahr=inf;
     while(fahr<=sup)
     {
       celsius = (5.0/9.0)*(fahr-32.0);
       cout<<"En fahrenheit : "<<fahr;
       cout<<" -  En celsius : "<<celsius<<endl;
       fahr = fahr+ pas;
     }
     // ----  2eme methode : boucle  dowhile  ----
     cout<<endl;
     fahr=inf;
     do
     {
       celsius = (5.0/9.0)*(fahr-32.0);
       cout<<"En fahrenheit : "<<fahr;
       cout<<" -  En celsius : "<<celsius<<endl;
     fahr = fahr+ pas;
     }
     while(fahr<=sup);
     // ----  3eme methode : boucle  for      ----
     cout<<endl;
     for(fahr=inf;fahr<=sup;fahr=fahr+pas)
     {
       celsius = (5.0/9.0)*(fahr-32.0);
       cout<<"En fahrenheit : "<<fahr;
       cout<<" -  En celsius : "<<celsius<<endl;
     }
     return 0;
}
```

Mise en page

Il est possible d'améliorer la mise en page comme proposé dans l'exemple suivant :

```
cout<<fixed;
cout.precision(2) ;
for(fahr=inf;fahr<=sup;fahr=fahr+pas)
{
  celsius = (5.0/9.0)*(fahr-32.0);
  cout<<"En fahrenheit : "<<setw(6)<<fahr;
  cout<<" -  En celsius : "<<setw(6)<<celsius<<endl;
}
```

Questions à se poser

- A quoi sert le setw(6) dans **cout<<setw(6)<<x ;** ? Quelle est la durée de l'effet de la commande ?
- A quoi sert **cout<<fixed ;** ? Quelle est la durée d'effet de cette commande ?
- A quoi sert **cout.precision(2);** ? Quelle est la durée de l'effet de la commande ?
- Pourquoi utilise-t-on la directive #include <iomanip> ?

Programme à réaliser

Ecrire un programme qui lit un entier N et qui affiche à l'écran une table des carres et des racines carrées de N premiers entiers.

NB. On peut réaliser a^x à l'aide de multiplications, ce qui est le plus rapide dans le cas ou x est entier. On peut aussi utiliser la fonction de bibliothèque pow(a,x). Pour pouvoir utiliser cette fonction, il faut inclure le fichier système "cmath" qui contient les prototypes des fonctions mathématiques. La fonction pow a des arguments de type double et une valeur de retour de type double.

III. SEQUENCE 3

Notions à maitriser

- Boucle de lecture de nombres

Exemple 3.1 : Programme de recherche du maximum d'une série d'entiers positifs

```
/*
     ********************************************************
     * Exemple 3.1 : Calcul du maximum d'une serie d'entiers *
     * positifs entres au clavier                            *
     ********************************************************
*/
#include <iostream>
using namespace std;
int main()
{
     int maxi=-1,i=1;
     cout<<"Entrez une serie d'entiers positifs"<<endl ;
     cout<<"tapez 0 pour terminer"<<endl;
     while(i>0)
     {
          cin>>i;
          if(i>maxi)maxi=i;
          cout<<"max provisoire : "<<maxi<<endl;
     }
     cout<<"Valeur du maximum : "<<maxi<<endl;
     return 0;
}
```

Points à noter

- Comme on ne traite que des entiers positifs, on peut prendre comme convention que l'entrée de 0 marque la fin de la série
- Pour initialiser la variable maxi, on peut prendre une valeur négative qui sera remplacée par le premier entier lu
- Pour que l'on entre dans la boucle il faut que i soit initialisé à une valeur positive, changée à la première lecture

Exemple 3.2 : Programme de recherche du maximum d'une série d'entiers quelconques

```
/*
     ********************************************************
     * Exemple3.2 : Calcul du maximum d'une serie d'entiers  *
     * quelconques entres au clavier                         *
     ********************************************************
*/
#include <iostream>
using namespace std;
```

```
int main()
{
      int maxi,i;
      cout<<"Entrez une serie d'entiers quelconques au clavier"<<endl ;
      cout<<"tapez CTRL_Z suivi de <Entree> pour terminer"<<endl;
      cin>>maxi;
      while(cin)
      {
            cin>>i;
            if(cin)
            {
                  if(i>maxi)maxi=i;
                  cout<<"max provisoire : "<<maxi<<endl;
            }
      }
      cout<<"Valeur du maximum : "<<maxi<<endl;
      return 0;
}
```

Points à noter

- Comme on lit des nombres positifs, négatifs ou nuls, on ne peut pas choisir un entier particulier comme marque de fin de série.
- On utilise le **signal** fin de fichier (**CTRL_Z** sous Windows ou **CTRL_D** sous Mac ou Linux) pour indiquer la fin de la série.
- La variable cin, qui représente le clavier, peut être testée pour savoir si le signal fin de fichier a été envoyé.
- On effectue une première lecture pour initialiser la variable maxi.

Questions à se poser

- Lorsque l'on tape CTRL_Z lors de l'exécution de l'instruction **cin>>i ;** que se passe-t-il pour i ?
- Pourquoi fait-on le test **if(cin)** ?

Programme à réaliser

Ecrire un programme qui lit une série d'entiers quelconques et détermine le maximum, le minimum et la valeur moyenne de la série.

IV. SEQUENCE 4

Notions à maitriser

- Vecteur mono dimensionnel

Exemple 4.1 : Stockage d'entiers positif dans un vecteur

```
/*
     ******************************************************************
     * Exemple 4.1 : Entrée d'une série d'entiers positifs au clavier *
     * et stockage dans un vecteur a taille predefinie                *
     ******************************************************************
*/
#include <iostream>
#include <vector>
using namespace std;
int main()
{
     int nb=0,x=1,i;
     vector<int> v(100);     // vecteur de taille 100
     while(x>0)
     {
          cout<<"Valeur suivante : ";
          cin>>x ;
          if(x>0)
          {
               v[nb]=x ; nb++;
          }
     }
     cout<<"Contenu du vecteur : ";
     for(i=0;i<nb;i++)
          cout<<v[i]<<" ";
     cout<<endl;
     cout<<"Taille du vecteur : "<<v.size()<<endl;
     cout<<"Place occupee : "<<nb<<endl ;
     return 0;
}
```

Points à noter

- Les nombres à lire sont positifs, on peut décider que la valeur 0 marquera la fin de la série.
- La longueur de la série est inconnue mais elle doit être inférieure à 101.

Questions à se poser

- Que signifie l'instruction **vector<int> v(100);** ?
- Que vaut ici **v.size()**, est-il différent de nb ?

Exemple 4.2 : Stockage d'une série d'entiers positifs dans un vecteur

```
/*
     ********************************************************************
     * Exemple 4.2 : Entrée d'entiers positifs au clavier et stockage *
     * dans un vecteur vide au depart. Agrandissement du vecteur a    *
     * chaque ajout                                                   *
     ********************************************************************
*/
#include <iostream>
#include <vector>
using namespace std;
int main()
{
     int x=1,i;
     vector<int> t;                        // vecteur vide
     while(x>0)
     {
     cout<<"Valeur suivante : ";
     cin>>x ;
     if(x>0) t.push_back(x);       // agrandissement et stockage
     }
     cout<<"Contenu du vecteur : ";
     for(i=0;i<t.size();i++)cout<<t[i]<<" ";
     cout<<endl;
     return 0;
}
```

Questions à se poser

- Que signifie l'instruction **vector<int> v,** quelle est la taille de v ?
- Que signifie **t.push_back(x)** ?
- Quelle est la valeur de **v.size()** au début, puis à la fin du programme ?

Programme à réaliser

Ecrire un programme qui lit une série d'entiers quelconques (positifs, négatifs ou nuls) au clavier. On ne connaît pas a priori la longueur de la série, c'est l'utilisateur qui décide d'arrêter de fournir des entiers en envoyant le signal **fin de fichier**. On calcule la somme, la moyenne et la variance.

NB. $\text{variance} = \dfrac{\sum_1^N (X_i - moyenne)^2}{\text{N}}$

Indication

On peut lire la série d'entiers et la stocker dans un vecteur. On peut calculer la moyenne pendant la lecture, puis on peut reprendre les valeurs stockées dans le vecteur pour calculer la variance.

V. SEQUENCE 5

Notions à maitriser

- Fonctions

Exemple 5.1 : Lecture de deux entiers, appel d'une fonction pour en calculer la somme, puis d'une fonction pour permuter leur valeur

```
/*
     ************************************************************
     * Exemple 5.1 : Fonctions de permutation avec arguments par  *
     * référence et par adresse                                   *
     ************************************************************
*/
#include <iostream>
using namespace std;

int somme(int,int);
void permutation_p(int*, int*);
void permutation_r(int&, int&);

int main()
{
    int x,y;
    cout << " Donner la valeur de x: ";
    cin >> x;
    cout << " Donner la valeur de y: ";
    cin >> y;

    cout << "Valeurs de x et y :" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;

    int s = somme(x,y);
    cout << "Valeur de " << x << " + " << y << ": " << s << endl;

    int *px, *py;
    px = &x;
    py = &y;
    permutation_p(px,py);
    cout << "Valeurs de x et y apres permutation :" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;

    permutation_r(x,y);
    cout << "Valeurs de x et y apres nouvelle permutation:" << endl;
    cout << "x = " << x << endl;
    cout << "y = " << y << endl;

    return 0;
}

int somme(int a, int b) // passage d'arguments par copie
{
    return a+b;
}

void permutation_p(int *a, int *b) // passage d'arguments par adresse
```

```
{
    int temp = *a;
    *a = *b;
    *b = temp;
}

void permutation_r(int &a, int &b) // passage d'arguments par référence
{
    int temp = a;
    a = b;
    b = temp;
}

// ----------------------------------------
```

Programme à réaliser

Reprendre le programme réalisé à la séquence 4, mais cette fois-ci, le calcul de la moyenne et de la variance sera écrit au sein d'une fonction prenant trois arguments :

- le vecteur d'entier,
- la moyenne calculée,
- la variance calculée.

Ecrire deux variantes de cette fonction :

- statistique_p : en utilisant un passage d'arguments par adresse (pointeurs)
- statistique_r : en utilisant un passage d'arguments par référence

VI. SEQUENCE 6

Notions à maitriser

- Variables pointeurs
- Allocation dynamique

Exemple 6.1 : Lecture au clavier d'un tableau d'entiers t, lecture d'un entier x, élimination des occurrences de x dans le tableau t en tassant les éléments restants.

```
/*
     ************************************************************
     * Exemple 6.1 : Elimination des occurrences d'une valeur   *
     * dans un tableau alloué dynamiquement et géré             *
     * avec des pointeurs                                       *
     ************************************************************
*/
#include <iostream>
using namespace std;

int main()
{
    int *p1, *p2;
    int x, i, n, *t, n2;

    cout << "Donner la dimension du tableau: ";
    cin >> n;
    t = new int[n];//Allocation dynamique d'un tableau à n valeurs entieres
    for (int i=0; i<n; i++)
    {
        cout << "Donner la valeur de t["<< i << "]: ";
        cin >> t[i];
    }

    cout << "Valeurs du tableau :" << endl;
    for (int i=0; i<n; i++)
    {
        cout << "t[" << i << "] = " << t[i] << endl;
    }

    cout << "Donner la valeur de l'element a supprimer : ";
    cin >> x;

    for(n2=0,p1=p2=t; p1<t+n; p1++)
    {
        if(*p1 != x)
         {
            *p2 = *p1;
            p2++;
            n2++;
         }
    }
    cout << "Nouvelles valeurs du tableau :" << endl;
    for(i=0,p1=t;p1<t+n2;p1++,i++)
    {
        cout << "t[" << i << "] = " << *p1 << endl;
    }
```

```
    return 0;
}
// ----------------------------------------
```

Programme à réaliser

- En utilisant les pointeurs et l'allocation dynamique, écrire un programme qui demande à l'utilisateur la taille n d'un tableau d'entier t,
- Remplit le tableau avec n valeurs saisies au clavier par l'utilisateur,
- Affiche le tableau t,
- Trie les données de t par ordre croissant (tri par bulles) à l'aide de pointeurs,
- Affiche le tableau t trié.

Indication : Principe du tri par bulles

Pour trier un tableau de n éléments, on effectue n parcours d'une "bulle" (2 éléments adjacents) de gauche à droite à travers le tableau. Lors d'un parcours on effectue la comparaison de 2 éléments successifs dans la bulle et on les permute s'ils ne sont pas classés correctement. Lors du premier parcours on a une bulle qui remonte de gauche à droite en entraînant le maximum. Lors du parcours suivant la bulle entraine le maximum des éléments restants, et ainsi de suite. En effectuant n parcours on obtient un tableau trié par valeurs croissantes.

Exemple d'un parcours

```
12   5  48   2   7   4        Données à trier


12   5  48   2   7   4
 5  12  48   2   7   4
 5  12  48   2   7   4
 5  12   2  48   7   4
 5  12   2   7  48   4
 5  12   2   7   4  48

 5  12   2   7   4  48
 5  12   2   7   4  48
 5   2  12   7   4  48
 5   2   7  12   4  48
 5   2   7   4  12  48
 5   2   7   4  12  48
```

etc.
Après 6 parcours de bulle le tableau sera ordonné.