semaine 5 Tutoriels
MOOC Init Prog C++

Les tutoriels sont des exercices qui reprennent des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même.

Ils sont conseillés comme un premier exercice sur un sujet que l'étudiant ne pense pas encore assez maîtriser pour aborder par lui-même un exercice « classique ».

Semaine 5 : Moyenne de classe (tableaux dynamiques)

> Cet exercice correspond à l'exercice « *pas à pas* » page 39 de l'ouvrage *C++ par la pratique* (3e édition, PPUR).

Nous allons dans cet exemple écrire un programme permettant à l'utilisateur de calculer la moyenne des notes d'une classe ainsi qu'afficher les écarts de chacun des élèves à cette moyenne.
Par exemple, nous souhaiterions avoir le déroulement suivant :

```
Saisie des notes :
  Entrez la note de l'élève 1 (<0 pour finir) : 4
  Entrez la note de l'élève 2 (<0 pour finir) : 4
  Entrez la note de l'élève 3 (<0 pour finir) : 5
  Entrez la note de l'élève 4 (<0 pour finir) : 6
  Entrez la note de l'élève 5 (<0 pour finir) : -2
Moyenne de classe : 4.75
Détail des notes :
 Elève : note (écart à la moyenne)
  1 : 4 (-0.75)
  2 : 4 (-0.75)
  3 : 5 (0.25)
  4 : 6 (1.25)
```

Comme on ne connaît pas *a priori* le nombre d'élèves (i.e. ce nombre n'est pas connu au moment où l'on écrit le programme, et pourrait même varier d'une exécution du programme à une autre), il est impossible de stocker la note de chaque élève dans une variable distincte. Nous avons besoin ici d'un tableau *dynamique* (`vector`).

Ecrivez le début du programme en y définissant un tableau dynamique pour stocker les notes des élèves.

(solution page suivante).

Solution :

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

int main()
{
  // déclare un tableau (dynamique) pouvant contenir
  // autant de notes que d'élèves
  vector<double> notes;

  return 0;
}
```

Maintenant que notre tableau est déclaré, nous pouvons le remplir en demandant à l'utilisateur la note de chaque élève.

Puisque le nombre d'élèves n'est pas connu *a priori*, on pourrait soit le demander à l'utilisateur, soit utiliser une convention pour indiquer la fin, par exemple une note négative. Utilisons ici cette seconde façon de procéder.

On en profite aussi pour calculer la somme des notes servant au calcul de la moyenne, au fur et à mesure qu'on les reçoit.

Ecrivez le code correspondant à la description ci-dessus.

(solution page suivante).

Solution :

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

int main()
{
  // déclare un tableau (dynamique) pouvant contenir
  // autant de notes que d'élèves
  vector<double> notes;
  double moyenne(0.0);

  // lecture des notes
  double note_lue(-1.0);
  cout << "Saisie des notes :" << endl;
  do {
    cout << "   Entrez la note de l'élève " << notes.size()+1
         << " (<0 pour finir) : ";
    cin >> note_lue;
    if (note_lue >= 0.0) {
       // stocke la note du nouvel élève
       notes.push_back(note_lue);

       // maintient à jour la somme des notes
       moyenne += note_lue;
    }
  } while (note_lue >= 0.0);

  return 0;
}
```

Il est maintenant possible de calculer la moyenne de classe, et d'afficher l'écart à la moyenne pour chaque élève. Ecrivez le code correspondant.

(solution page suivante).

Solution :

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

int main()
{
  // déclare un tableau (dynamique) pouvant contenir
  // autant de notes que d'élèves
  vector<double> notes;
  double moyenne(0.0);

  // lecture des notes
  double note_lue(-1.0);
  cout << "Saisie des notes :" << endl;
  do {
    cout << "   Entrez la note de l'élève " << notes.size()+1
         << " (<0 pour finir) : ";
    cin >> note_lue;
    if (note_lue >= 0.0) {
       // stocke la note du nouvel élève
       notes.push_back(note_lue);

       // maintient la somme à jour
       moyenne += note_lue;
    }
  } while (note_lue >= 0.0);

  moyenne = moyenne / notes.size();
  /* Attention ici ! notes.size() est de type entier. Comme nous avons
   * bien fait de déclarer moyenne comme double, la division est bien la
   * division usuelle que nous souhaitons. (ouf !)
   * Mais si nous avions déclaré moyenne comme int, la division serait alors
   * la division entière et le résultat serait sûrement faux (pas de partie
   * fractionnaire, ce qui correspond bien à un int ! ;-) ).
   * Il faut donc faire bien attention à utiliser les types dont on a besoin.
   */

  // affichage des résultats
  cout << "Moyenne de classe : " << moyenne << endl;
  cout << "Détail des notes : " << endl;
  cout << " Elève : note (écart à la moyenne)" << endl;
  for (size_t i(0); i < notes.size(); ++i)
  {
    cout << "  " << i+1 << " : " << notes[i] << " ("
         << (notes[i] - moyenne) << ")" << endl;
  }

  return 0;
}
```

Complément C++11 : depuis la nouvelle norme 2011, on peut utiliser une autre sorte d'itération, et remplacer

```
for (size_t i(0); i < notes.size(); ++i)
{
  cout << "  " << i+1 << " : " << notes[i] << " ("
       << (notes[i] - moyenne) << ")" << endl;
}
```

par

```
int i(1);
for (auto note : notes)
{
  cout << "  " << i << " : " << note << " ("
       << (notes - moyenne) << ")" << endl;
  i = i + 1;
}
```

mais l'intérêt est ici limité vu que l'on a besoin du numéro `i` de l'élève. Cette sorte d'itération (C++11) prend tout son sens lorsque que l'on veut parcourir le tableau *sans* avoir besoin explicitement des numéros de positions (i.e. des *index*).
