Université Paris-XII – Faculté de Sciences économiques et de Gestion
Master Professionnel *Méthodes Appliquées de la Statistique et de l'Économétrie pour la Recherche, l'Analyse et le Traitement de l'Information*
Année universitaire 2009-2010 – *François Legendre*
F.Legendre@univ-paris12.fr – http://fj.legendre.free.fr

Mini-projet du troisième jour de cours

Le mini-projet réalisé au cours du troisième jour de formation poursuivait le double but suivant

- concevoir un objet qui représenterait un étudiant – son nom et la série de notes que ce dernier aurait obtenues ;
- réaliser un traitement élémentaire qui utiliserait cet objet.

Le nom des étudiants est enregistré dans un fichier (de nom « `maserati.txt` ») à raison d'un nom par ligne. Les notes sont enregistrées dans un fichier (de nom « `notes.txt` ») ; les étudiants peuvent avoir une ou plusieurs notes, c'est une erreur si un étudiant n'a aucune note. Il s'agit de produire un état synthétique comportant les trois colonnes suivantes

1. le nom ;
2. le nombre de notes ;
3. la moyenne obtenue par l'étudiant.

Le programme ci-après détaille la solution construite en cours.

```
#include <iostream> // Flux cin, cout et cerr.
#include <fstream> // Files stream.
#include <string>  // string de la STL.
#include <vector>  // vector de la STL.
#include <cstdlib> // Bibliothèque standard du langage C.

using namespace std ;

class Etudiant {
  // Attributs de la classe.
  string nom ;
  vector <double > notes ;
public :
  // Constructeur de la classe.
  Etudiant (const string & arg) : nom(arg), notes() {}

  // Méthode pour ajouter une nouvelle note.
  void nouvelle_note(double arg)
  {
    if ( (arg < 0) || (arg > 20) )
    {
      cerr << "La note est invalide." << endl ;
      exit(1) ;
    }
    notes.push_back(arg) ;
  }
  // Méthode pour obtenir le nombre de notes de l'étudiant.
  size_t nombre_notes() const
  {
    return notes.size() ;
  }
  // Méthode pour calculer la moyenne de l'étudiant.
  double moyenne() const
  {
    if ( notes.size() == 0 )
    {
      cerr << "Oups, l'étudiant n'a pas de note !" ;
      exit(1) ;
    }
    double somme = 0 ;
    for ( size_t i = 0 ; i < notes.size() ; ++i )
      somme += notes[i] ;
    return somme / notes.size() ;
  }
  string get_nom() const
  {
    return nom ;
  }
} ;

// Cette fonction permet de peupler un 'vector<Etudiant>' en lisant le fichier
// dont le nom est passé en argument. Ce fichier comporte le nom des étudiants
// à raison d'un nom par ligne. Cette fonction retourne le 'vector<Etudiant>'
// convenablement peuplé.
vector<Etudiant> lire(const string & fichier)
{
  // Un 'vector<Etudiant>' temporaire, vide, est créé.
  vector<Etudiant> tmp ;
  // Un input file stream est ouvert (en lecture, bien sûr).
  ifstream in( fichier.c_str()) ;
  if ( ! in )
  {
    cerr << "Oups, ouverture impossible de '" << fichier << "'." << endl ;
```

```
64      exit(1) ;
65    }
66    string ligne ;
67    // Le vector est peuplé en recopiant un 'Etudiant' qui a été construit à
68    // partir du nom lu dans le fichier externe.
69    while ( in >> ligne ) // Tant que la fin de fichier n'a pas été atteinte.
70      tmp.push_back(Etudiant(ligne)) ;
71
72    cout << "Lecture de " << fichier << " achevée." << endl ;
73    cout << "Il y a " << tmp.size() << " étudiants." << endl ;
74    return tmp ;
75  }
76
77  // Point d'entrée dans le programme.
78  int main () {
79
80    // Déclaration et initialisation de 'maserati' le vector dont les éléments
81    // sont les étudiants. leur nom est lu dans le fichier 'maserati.txt'. On
82    // aurait pu coder, pour reprendre une syntaxe orientée objet :
83    //    vector<Etudiant> maserati( lire("maserati.txt") ) ;
84    vector<Etudiant> maserati = lire("maserati.txt") ;
85
86    // Le fichier comporte, sur chacune de ses lignes, le nom d'un étudiant et
87    // la note obtenue à une épreuve. Le nombre de notes par étudiant est a
88    // priori variable.
89    ifstream in("notes.txt") ;
90    if ( ! in )
91    {
92      cerr << "Oups, ouverture impossible de 'notes.txt'." << endl ;
93      exit(1) ;
94    }
95    string nom_lu ;    // Nom de l'étudiant, lu dans le fichier 'notes.txt'.
96    double note_lue ;  // Note de l'étudiant, lue dans ce même fichier.
97    while ( in >> nom_lu >> note_lue )
98    {
99      // Recherche séquentielle dans le vector 'maserati' pour trouver l'élé-
100     // ment de ce vector dont le nom correspond. La variable booléenne 'ok'
101     // sera égale à true si l'étudiant a été trouvé.
102     bool ok = false ;
103     for ( size_t i = 0 ; i < maserati.size() ; ++i )
104     {
105       if ( maserati[i].get_nom() == nom_lu )
106       {
107         // L'étudiant a été trouvé. On lui attribue une nouvelle note.
108         maserati[i].nouvelle_note(note_lue) ;
109         ok = true ;
110         break ;
111       }
112     }
113     if ( ! ok )
114     {
115       cerr << "Oups, l'étudiant n'existe pas." << endl ;
116       exit(1) ;
117     }
118   }
119   cout << "Lecture de 'notes.txt' achevée." << endl ;
120   cout << "Nom\t# de notes\tMoyenne" << endl ;
121   for ( size_t i = 0 ; i < maserati.size() ; ++i )
122     cout << maserati[i].get_nom() << '\t' <<
123         maserati[i].nombre_notes() << '\t' <<
124         maserati[i].moyenne() << endl ;
125
126   return 0 ;
127 }
```

Le programme C++ qui se limite à l'exploitation du fichier des notes est beaucoup plus simple. Il figure ci-après.

```
1   #include <iostream> // Flux cin, cout et cerr.
2   #include <fstream> // Files stream.
3   #include <string>  // string de la STL.
4   #include <map>     // vector de la STL.
5   #include <cstdlib> // Bibliothèque standard du langage C.
6
7   using namespace std ;
8
9   // Point d'entrée dans le programme.
10  int main () {
11
12    // Correspondance dont la clef est le nom et la valeur la paire (n, somme).
13    map< string, pair<size_t, double> > dict ;
14
15    ifstream in("notes.txt") ;
16    if ( ! in )
17    {
18      cerr << "Oups, ouverture impossible de 'notes.txt'." << endl ;
19      exit(1) ;
20    }
```

```
21    string nom ; // Nom de l'étudiant, lu dans le fichier 'notes.txt'.
22    double note ; // Note de l'étudiant, lue dans ce même fichier.
23    while ( in >> nom >> note )
24      dict[nom] = make_pair(dict[nom].first+1, dict[nom].second+note) ;
25
26    typedef map< string, pair<size_t, double> >::const_iterator T ;
27    for ( T it = dict.begin() ; it != dict.end() ; ++ it )
28    {
29      size_t n = it->second.first ;
30      double somme = it->second.second ;
31      cout << it->first << '\t' << n << '\t' << (somme/n) << endl ;
32    }
33    return 0 ;
34  }
```