TP8 (2 séances)

Objectif : programmer une classe « `commande` » permettant d'établir des statistiques sur les commandes client de la société Gizmo.

Détails des fonctionnalités :

Enregistrer les commandes à partir du fichier.
Compter le nombre de clients distincts.
Calculez le volume global des commandes.
Afficher la liste des clients par ordre croissant (sur leur numéro), indiquez pour chaque client son nombre de commandes et le montant global de ses commandes (somme des montants de ses commandes).
Afficher la liste des clients d'après leur importance pour la société Gizmo. C'est à dire en premier, les clients qui ont un montant global plus important.
Calculer le nombre moyenne de commandes par client
Calculer le « montant global » moyen par client
Stocker le résultat de l'analyse dans un fichier.

Première étape : enregistrer les commandes stockées dans un fichier. Le nom du fichier est indiqué sur la ligne de commande (paramètre de l'exécutable).

La structure du fichier de données est la suivante : une ligne par commande. Chaque ligne a la structure suivante : `numCommande numClient montant`

Un exemple est le fichier `/net/exemples/AS/TP_ASD_PROG/donnees-tp8` qui contient moins de 20 commandes.

Un exemple de code pour saisir des données complexes stockées dans un fichier (nommé `data`) est en annexe.

Algorithmique

1. Complétez la structure statique donnée ci-dessous permettant de manipuler au plus `NB_MAX_COMMANDES` commandes - n'hésitez pas à concevoir de nouvelle classe si besoin -
2. Ecrivez un algorithme affichant sur la sortie standard la liste des clients par ordre croissant de numéro.
3. Ecrivez un algorithme affichant sur la sortie standard la liste des clients dans l'ordre d'importance pour Gizmo.
4. Proposez un format de stockage des résultats dans un fichier.

Programmation

1. Complétez les fichiers d'entête « `Commande.h` » et « `TabCommandes.h` » contenant respectivement les déclarations des constructeurs, des destructeurs, des méthodes de la classe `Commande` et de la classe `TabCommandes`, ainsi que l'ensemble de leurs membres.
2. Ecrivez le code de la classe `Commande` et de la classe `TabCommandes`.

<u>Dossier à rendre début janvier 2011 :</u>

1. Présentation de votre structure de données
2. L'algorithme affichant sur la sortie standard la liste la liste des clients par ordre croissant sur leur identifiant
3. L'algorithme affichant sur la sortie standard la liste des mots dans l'ordre d'importance, avec leur nombre d'occurrences

4. Le code facilement lisible (attention aux noms des méthodes, aux noms des variables, aux commentaires sur les méthodes, ...).
5. Jeu de tests que vous avez utilisé pour tester et valider votre code.
6. Un état honnête d'avancement du projet (précisant les fonctionnalités qui n'ont pas été implémentées, si vous avez détecté des bugs, ...).

Optionnel :

1. **Structure dynamique-** Concevez et programmez la classe « `DynamiqueTabCommandes` » permettant d'analyser des textes de taille quelconque. Vous pouvez, si vous le souhaitez, utiliser la STL (standard library of C++), mais cela n'est pas obligatoire.

```
============================
     Commande.h
============================
#include <iostream>
using namespace std;

class Commande {
private:
  int _numeroCommmande;
  int _numeroClient;
  float _cout ;
public :
  Commande();
  Commande(int numCmd, int numClient, float cout)  ;
  int getNumClient() ;
  float getCout() ;
  int getNumCommande() ;
  void afficher();
};
```

```
============================
     TabCommandes.h
============================
#include <fstream>
#include <cstdlib>
#include <Commande.h>

#define NB_MAX_COMMANDES 256

class TabCommandes {
private :
    int _taille;
    Commande _tabCmd[NB_MAX_COMMANDES];
public :
    TabCommandes(ifstream &ficEntree) ;
    int nombreClient() ;
    float montantGlobal() ;
    void afficherOrdreNumerique () ;
    void afficherImportance () ;
    void afficher();
    int getTaille() ;
    Commande getCommande(int position) ;
};
```

==========
Exemple de lecture de données structurées stockées dans un fichier.
==========

```
#include <stdio.h>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
using namespace std;

int main( int argc, const char* argv[] )
{
     int c1;
     float f1;
     string line;
     istringstream instream;
     ifstream in("data");
     if(!in) { cerr << "Cannot open input file.\n"; return 1;}
     while(!in.eof()) {
       getline(in,line);
       instream.clear();     // Reset from possible previous errors.
       instream.str(line);      // Use "line" as source of input.
       if (instream >> c1 >> f1) {cout << c1 << ", " << f1 << endl;}
       else { cout << "bad format" << endl;     }
     }
     in.close();
     return 0;
}
```