TP8 (3 séances)

Adapté du cours de « Michel Billaud »

Objectif : programmer une classe « `CarnetOrdres` » permettant d'analyse le carnet d'ordres d'une action sur une journée à la bourse de JolieVille.
Chaque matin, les autorités de la bourse JolieVille établissent le cours des actions à partir du « carnet d'ordres » de la veille**.**
Le cours d'une action est le montant minimal qui permet de réaliser un maximum de transactions. Le volume des transactions à un cours donné est le minimum des deux valeurs suivantes :

- volume maximum d'achats possibles
- volume maximum de ventes possibles

remarque : un client peut donner plusieurs ordres durant la même journée.

Carnet d'ordres de l'action « AAA » du 5 septembre 2011:

Numéro client	Type ordre	quantité	cours
5	a	150	2,1€
2	v	50	2€
4	v	80	3€
3	a	90	1€
1	v	20	1,95€
2	v	30	1,9€
6	a	10	2€
2	v	10	2,5€

Explication du carnet d'ordres de l'action « AAA » du 5 septembre 2011 :

- L'ordre du client 3 est l'achat de 90 actions pour un coût par action d'au plus de 1€.
- L'ordre du client 4 est la vente de 80 actions, pour un prix par action d'au moins 3€.

L'analyse du carnet d'ordres est détaillée dans le tableau suivant où les ordres sont classés par ordre décroissant sur les cours. Le cours de l'action « AAA », le 5 septembre 2011, est de 2€ le volume des transactions est de 100 actions.
Les ordres d'achats à un cours inférieur à 2 euros ne sont pas réalisés ; les ordres de vente à un cours supérieur à deux euros ne sont pas réalisés.

Numéro client	Type ordre	quantité	cours	Volume maximum achats au cours courant	Volume maximum vente au cours courant	Volume maximum des transactions	Niveau réalisation de l'ordre ?
4	v	80	3€	0	190	0	0
2	v	10	2,5€	0	110	0	0
5	a	150	2,1€	150	100	100	100
6	a	10	2€	160	100	100	0
2	v	50	**2€**	160	100	**100**	50
1	v	20	1,95€	160	50	50	20
2	v	30	1,9€	160	30	30	30
3	a	90	1€	250	0	0	0

Le résultat des transactions correspondant au carnet d'ordres « AAA » est :

numéro client	Nature transaction	montant	quantité
1	c	+40€	-20
2	c	+160€	-80
3	r	0€	0
4	r	0€	0
5	d	-200€	+100
6	r	0€	0

Détails des fonctionnalités (voir la structure du programme ci-joint):

1. Lire le « carnet d'ordre » d'une action stocké dans un fichier. Le nom du fichier est donné sur la ligne de commande (paramètre de l'exécutable).
   La structure du fichier de données est la suivante : une ligne par ordre. Chaque ligne a la structure suivante : `numero_client type_ordre quantité cours.`
   N'oubliez pas de vérifier les données, par exemple le type d'un ordre est « a » ou « v ».
2. Calculer le cours de l'action (le cours minimal qui permet de réaliser un maximum d'échanges) et le volume des transactions.
3. Calculer et afficher le résultat des transactions. Pour chaque client, est calculé :
   - la nature de la transaction finale (crédit/vente, débit/achat ou rien) ;
   - le montant qu'il doit recevoir ou qu'il doit payer ;
   - la quantité d'actions qu'il a vendues ou achetées.

Algorithmique

1. Ecrivez un algorithme calculant le cours de l'action et le montant des échanges.
2. Ecrivez un algorithme calculant pour chaque client les résultats des transactions

<u>Le dossier à rendre début janvier 2012, doit contenir :</u>

1. la présentation des deux algorithmes
2. le code facilement lisible (attention aux noms des méthodes, aux noms des variables, aux commentaires sur les méthodes, ...). Le code doit correspondre aux algorithmes que vous avez écrits. N'hésitez pas à compléter les fichiers .h donnés en annexe.
3. le jeu de tests que vous avez utilisé pour tester votre code.
4. un état honnête d'avancement du projet (précisant les fonctionnalités qui n'ont pas été implémentées, si vous avez détecté des bugs, ...).

<u>Optionnel :</u>

1. **Structure dynamique-** Concevez et programmez la classe « `DynamiqueCarnetOrdres` » permettant d'analyser un carnet d'ordres de taille quelconque. Vous pouvez, si vous le souhaitez, utiliser la STL (standard library of C++).

```
============================
      Ordre.h
============================
//TP8 AS – 2011-2012
// auteur : Colette Johnen

#include <iostream>
using namespace std;

class Ordre {
private :
  int _numeroOrdre;
  int _numeroClient;
  char _typeOrdre;
  int _quantité;
  float _cours;

public :
  Ordre();
  Ordre(int numeroOrdre, int numClient, char typeOrdre, int quantite, float cours);
  int getNumeroOrdre() ;
  int getNumeroClient() ;
  float getCours() ;
  char getTypeOrdre() ;
  int getQuantite();
  void afficher();
};


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

class TransactionClient {
private :
  int _numeroClient;
  char _natureTransaction;
  int _quantité ;
  float _montant ;
public :
  TransactionClient();
  int getNumeroClient() ;
  float getMontant() ;
  char getTypeTransaction() ;
  int getQuantite();
  void afficher();
};
```

```
============================
     CarnetOrdres.h
============================
#include <fstream>
#include <cstdlib>
#include <Ordre.h>
#include < TransactionClient.h>

#define NB_MAX 256

class CarnetOrdres {
private :
    int _tailleOrdre;
    Ordre _tabOrdres[NB_MAX] ;

    float _coursAction ;
    int _volumeTransaction ;

    int _tailleTransactionClient;
    TransactionClient _tabTransactionClients[NB_MAX];


public :
    CarnetOrdres(ifstream &ficEntree) ;
    void afficherOrdres();

    void calculerCoursAction () ;
    void afficherCours() ;

    void calculerTransactionClient() ;
    void afficherTransactionClient() ;

};
```