Projet 1 : Magasins (2 séances, à rendre)

Ce projet (et les suivants) visent à faire implémenter une simulation de super-marché dans lequel un client remplit un chariot (caddie™) avec des produits puis passe en caisse pour payer. Dans un premier temps, on s'intéresse à modéliser le magasin : il vend des produits dont certains sont en stock, et pour chaque produit, un prix est fixé.

Un premier travail a été effectué et une implémentation vous est fournie mais utilisant des structures de la Bibliothèque standard (STL). Vous pouvez déjà compiler ce code, il fonctionne (normalement).

Votre travail consiste à remplacer les structures de Table Associative utilisées par les vôtres et à comparer les coûts correspondants.

Implémentation fournie :

Vous trouverez sur Madoc une archive contenant :

- une classe Produit **à ne pas modifier** (`produit.hpp` + `produit.cpp`) implémentant les produits vendus dans le magasin ;
- une classe Mappe **à ne pas modifier** (`mappe.hpp` + `mappe.tpp`) implémentant une table associative en utilisant la STL ;
- un programme de test de cette classe (`TA_test.cpp`) que vous compléterez pour vérifier que vos classes (AList et Htable) se comportent de la même façon que celle fournie ;
- une classe Magasin à compléter (`magasin.hpp` — **à ne pas modifier** — + `magasin.tpp`) prenant en paramètre la classe implémentant la table associative ;
- un programme de test de cette classe (`magasin_test.cpp`) que vous compléterez pour vérifier que le magasin se comporte de la même façon avec les deux implémentations (celle fournie et la vôtre) ;
- un programme exécutable (`entreposer`) permettant de générer aléatoirement des fichiers de produits pour utiliser le constructeur de Magasin ; un exemple de fichier généré est aussi fourni (`entrepot_11.txt`) ;
- un makefile pour compiler (utiliser `make all` dans le dossier).

Travail à effectuer :

Commencez par prendre connaissance du contexte : lisez intégralement le sujet (;-) puis étudiez le code fourni. Compilez-le et exécutez pour vous l'approprier. Profitez-en pour noter dans un document de travail les questions que vous vous posez, les remarques qui vous viennent à l'esprit, les idées d'amélioration, tout cela (une partie tout au moins) nourrira votre rapport. Ce document présentera aussi les choix effectués (constantes, comportement des méthodes non précisé dans l'énoncé, …) et les limites d'utilisation. Il est aussi, si ce n'est plus important, que le code.

Adaptez le code de vos classes `AListe` et `HTable` pour qu'elles respectent la spécification de `Mappe` puis testez les dans TA_test. Compléter TA_test pour mieux les vérifier.

Complétez le code de la classe Magasin : il manque dans `magasin.ttp` la définition des fonctions `tarif`, `stock`, `vendre`, `solder`, `inventaire`, `capital` et `nettoyageDePrintemps`.

Modifiez le programme de test de magasin pour les vérifier avec la classe Mappe fournie puis en appelant vôtre structure (il suffit de changer la classe passée en paramètre du constructeur) ; complétez-le.

Comparez les temps d'exécution de vos structures avec celle fournie, sur des exemples pertinents ; faites un graphique. Vous devriez pouvoir faire mieux que la STL !

Documents à rendre :

Une archive contenant le code C++ utilisé et le rapport.

- Pour le code, vous pouvez simplement créer une archive du dossier projet1 complet (supprimer les exécutables pour gagner de la place). Il doit y figurer le code de vos classes (AListe.hpp, AListe.tpp, HTable.hpp, HTable.tpp) et de celle complétée (magasin.tpp) ainsi que des programme de test (TA_test.cpp, magasin_test.cpp).
- Le rapport, impérativement au format pdf, devra comporter un comparatif des implémentations en terme de complexité temporelle (l'ordre de grandeur du coût de chaque méthode doit être renseigné) et si possible d'encombrement mémoire. D'autres implémentations (suffisamment différentes) peuvent y être présentées (la réalisation n'en est pas demandée) et comparées.

Le travail est à déposer sur MADOC au plus tard le 3 mars 2014 à 23h55 sous forme d'une archive zip contenant le rapport (pdf) et le code (archive du dossier complet) .

Le code doit être clair et concis, commenté, avec en particulier les pré- et post-conditions. Le rapport doit être bien écrit (français correct, attention à l'orthographe !) et agréable à lire, il doit comporter introduction et conclusion et des annexes pour les détails techniques (graphiques, code, etc.)

Vous pouvez proposer (sans les implémenter) quelques méthodes supplémentaires si leur coûts sont très différents hors et dans l'implémentation. Elles seront peut-être reprises pour la suite du projet.