INF1010

Programmation Orientée-Objet

Travail pratique #5

Classes génériques et la STL

Objectifs :	Permettre à l'étudiant de se familiariser avec le concept de fonctions et des classes génériques, aux foncteurs, aux conteneurs et algorithmes de la STL
Remise du travail :	Lundi 8 Avril 2018, 8h
Références :	Notes de cours sur Moodle & Chapitres 11 à 14, 16 et 20 du livre Big C++ 2e éd.
Documents à remettre :	Les fichiers .cpp et .h complétés, les questions en pdf le tout sous la forme d'une archive au format .zip.
Directives :	Directives de remise des Travaux pratiques sur Moodle Les travaux dirigés s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe. Veuillez suivre le guide de codage

Informations préalables

Veuillez lire attentivement tout l'énoncé avant de commencer à écrire du code, afin de vous assurer de bien comprendre l'interaction entre les classes. Cela vous permettra aussi de commencer par ce qui vous semble le plus pertinent, car en effet l'ordre de description des classes dans le TP n'est pas forcément l'ordre à suivre. Il pourrait par exemple être plus intéressant de faire toutes les méthodes liées à l'insertion d'une donnée pour pouvoir les tester et continuer par la suite.
Assurez-vous aussi de bien comprendre le fonctionnement des templates et leur utilité pour créer des classes génériques. En complément au cours et la documentation de la STL, vous pouvez consulter ce lien.
Le travail effectué dans ce TP continue celui amorcé par les TP 1, 2, 3 et 4, soit une simulation de restaurant en y intégrant les notions de foncteurs, classes génériques, les conteneurs et les algorithmes de la STL.
Les foncteurs sont des objets qui peuvent agir comme des fonctions grâce à la surcharge de l'opérateur « () ». Les foncteurs sont très utiles lorsqu'on travaille avec des structures de données (conteneurs) provenant de la bibliothèque STL comme une list ou une map...
Les foncteurs peuvent avoir aucun ou plusieurs attributs. Généralement, un foncteur sans attribut permet de manipuler les objets sur lesquels il est appliqué. Les foncteurs à un attribut sont souvent utilisés à des fins de comparaison, mais aussi d'autres utilités.
L'utilisation du **static_cast** est **INTERDITE**. Tout le TP est réalisable sans cette fonction.
Pour vous aider, les fichiers du TP précédent vous sont fournis. Vous n'avez qu'à implémenter les nouvelles méthodes décrites plus bas. Les attributs ou méthodes qui ne sont plus nécessaires, ont été supprimés. Et les méthodes à modifier vous ont été indiquées.
ATTENTION : Tout au long du TP, assurez-vous d'utiliser les opérateurs sur les objets et non sur leurs pointeurs ! Vous devez donc déréférencer les pointeurs si nécessaires.
ATTENTION : Vous serez pénalisés pour les utilisations inutiles du mot-clé *this*. Utilisez-le seulement où nécessaire.
ATTENTION : Il est fortement recommandé d'utiliser les fichiers fournis, plutôt que de continuer avec vos fichiers du TP4.
ATTENTION : L'utilisation de boucles for ou while de la forme for (int i; i <vec.size(); i++) est interdit pour les nouvelles méthodes que vous devez implémenter. Vous devez soit utiliser les algorithmes lorsque possible, soit utiliser les boucles for/while en utilisant les itérateurs.
Remarque : Pour plus de précision sur le travail à faire et les changements à effectuer, veuillez-vous référer aux fichiers .h.

Travail à réaliser

Lisez la description des classes ci-dessous et suivez aussi les indications des **TODO** insérées dans le code pour apporter vos modifications au code fourni.
Si une méthode d'affichage ne respecte le format de la capture donnée en annexe, vous devez modifier les méthodes en question.

Classes GestionnaireGenerique

Cette classe est une classe générique pour différents types de gestionnaires. Sa définition vous est donnée dans le fichier GestionnaireGenerique.h. Vous devez implémenter les méthodes suivantes :

- void ajouter(T t) : permet d'ajouter un élément au conteneur conteneur_.
- int getNombreElements() const : retourne la taille du conteneur_.
- C getConteneur() const : retourne une copie du conteneur_.

Classe GestionairePlats

La classe GestionnairePlats dérive de la classe GestionnaireGenerique. Le conteneur sous-jacent doit être un map entre le nom des plats et les pointeurs vers ces plats. Vous devez trouver ce que sont les éléments contenus dans un map pour que la classe de base soit cohérente.
Vous devez la créer et implémenter les méthodes suivantes :

- GestionnairePlats(const string& nomFichier, TypeMenu type) :
  Créer un GestionnairePlats à partir d'un nom de fichier le type de menu qui sera contenu. Cette méthode doit appeler la méthode lirePlats.

- GestionnairePlats(GestionnairePlats* gestionnaire) :
  Contructeur par copie. Elle crée un GestionnairePlats à partir de l'adresse d'un autre en ajoutant les plats au nouveau GestionnairePlats.

- ~GestionnairePlats() : Destructeur de GestionnairePlats. Il faut désallouer les plats du gestionnaire sous-jacent.

- TypeMenu getType() const : retourne le type de Menu gérer.

- Plat* allouerPlat(Plat*) : crée un plat à partir d'un autre.

- Plat* trouverPlatMoinsCher() const :
  En utilisant un algorithme de la STL et le foncteur FoncteurPlatMoinsCher, vous devez retourner le plat le moins cher du conteneur.

- Plat* trouverPlatPlusCher() const : En utilisant une fonction lambda et un algorithme de la STL, vous devez trouver le plat le plus cher du conteneur.

- Plat* trouverPlat(const string& nom) const: Retourne le plat du conteneur qui porte le nom spécifié.

- vector<pair<string, Plat*>> getPlatsEntre(double borneInf, double borneSup): En utilisant le FoncteurIntervalle, retournez les plats dont les prix sont compris entre la borneInf et la borneSup.

- void afficherPlats(ostream& os) : Méthode qui permet d'afficher les plats du conteneur.

Classe GestionnaireTables

La classe GestionnaireTables dérive de la classe GestionnaireGenerique. Le conteneur sous-jacent doit être un conteneur set de pointeurs à des objets tables.

Vous devez implémenter les méthodes suivantes :

- Table* getTable(int id) const: retourne la table dans le conteneur qui porte le id spécifié.
- Table* getMeilleureTable(int tailleGroupe) const : retourne la plus petite table pouvant contenir le groupe.
- void afficherTables(ostream& os) const : affiche les tables en utilisant des itérateurs.

Foncteur.h

Vous devez implémenter les différents Foncteurs qui sont utilisés par la classe GestionnairePlats :

- FoncteurPlatMoinsCher : Un foncteur prédicat binaire permet de comparer les plats deux à deux du conteneur map.
- FoncteurIntervalle : Un foncteur prédicat unaire ayant la borne inférieure et la borne supérieure comme attributs. Ce foncteur prend en entrée un plat unique du conteneur map et vérifie si son prix est compris entre les bornes.

Classe Restaurant

Vous devez modifier Restaurant.cpp et Restaurant.h pour utiliser les classes GestionnairePlats et GestionnaireTables. Veuillez-vous fier aux commentaires dans les fichiers.

Questions

1. Aurait-il été possible de créer un autre gestionnaire pour les autres classes? Si oui, laquelle et quel type de conteneur conseilleriez-vous ?

2. Pourquoi l'implémentation des classes génériques est dans .h et non pas séparée en .h et .cpp comme les classes normales ?

Corrections

La correction du TP se fera sur 20 points.
Voici les détails de la correction :

- (4 points) Compilation du programme ;
- (3 points) Exécution du programme ;
- (5 points) Comportement exact des méthodes du programme et l'utilisation de la STL;
- (4 points) Utilisation adéquate des foncteurs;
- (1 points) Documentation du code ;
- (2 points) Allocation / désallocation appropriée mémoire
- (1 point) Réponse aux questions.

Affichage attendu

```
Erreur : table vide ou plat introuvable.

TESTS
        Test 01... OK!
        Test 02... OK!
        Test 03... OK!
        Test 04... OK!
        Test 05... OK!
        Test 06... OK!
        Test 07... OK!
        Test 08... OK!
        Test 09... OK!
        Test 10... OK!
        Test 11... OK!
        Test 12... OK!
        Test 13... OK!
        Test 14... OK!
        Test 15... OK!
        Test 16... OK!
        Test 17... OK!
        Test 18... OK!
        Test 19... OK!
        Test 20... OK!
        Test 21... OK!
        Test 22... OK!
        Test 23... OK!
        Test 24... OK!
```