IFT1166 - TRAVAIL PRATIQUE #3 - 15 Novembre 1999

Liste, Templates, et Héritage simple

Mohamed Lokbani

LA DATE LIMITE EST LE **Mercredi 08 Décembre 1999 MIDI**, avec une pénalité de 2 points par jours de retard.

À faire seul ou en équipe de deux.

BUT

Vous faire pratiquer la manipulation des patrons de fonctions ou de classes (Templates) et les listes chaînées. Un exemple avec l'héritage simple servira de locomotive pour ce TP.

Lisez l'énoncé de A à Z, puis en cas de problèmes contactez nous.

1. Présentation du problème

Pour manipuler une liste de données pour de types réels, entiers, complexes, rationnels, des comptes etc. ; il est possible d'écrire pour chacun des types cités, une classe Liste. On peut dire que c'est une opération fastidieuse, peu rentable en temps. Le langage C++ permet de créer des classes génériques dont le but et de compacter le code nécessaire pour décrire les types de données qu'une liste devra manipuler. Une classe générique appelée aussi un patron de classe, est un modèle que le compilateur utilise pour créer des classes au besoin. Un patron de classes permet de créer des classes générales et de transmettre des types comme paramètres à ces classes pour construire une classe spécifique.

La classe Liste n'a pas à se soucier du type réel de ses éléments, elle ne doit s'occuper que de réaliser certaines opérations sur ses éléments (trier les éléments ; supprimer, ajouter un élément etc.), on a donc intérêt à ne définir la classe Liste que par le type des éléments qu'elle manipule.

Le but de ce TP est de vous faire réaliser la classe Liste en vous inspirant de la démonstration #7, Et d'utiliser cette classe sur un exemple classique d'héritage simple.

2. Travail à réaliser

2.1 La Liste générique

-1- Écrire l'ensemble des fonctions membres décrites dans les classes génériques "**List**" et "**Node**" définies dans le fichier entête de la démonstration #7: "list.h"

Il ne vous est pas demandé de suivre cette déclaration à la lettre, vous pouvez en particulier ajouter ou retirer des membres. Seules les fonctionnalités offertes dans la déclaration doivent être respectées.

-2- Écrire le code nécessaire pour traiter le cas d'un nœud du type <char *>.

Vous devez écrire l'ensemble de ces déclarations dans un seul fichier qui portera le nom de: liste.h

2.2 Héritage simple

Soit la représentation suivante:

Une personne est connue par son nom et son prénom. Un étudiant est une personne a qui on associe une clé unique pour l'identifier. Un enseignant est lui aussi une personne, qui possède une clé unique pour l'identifier. Un étudiant peut prendre un ou plusieurs cours, un professeur quant à lui peut donner un ou plusieurs cours. Quand au cours, il est identifiable par une clé unique et peut être pris par un ou plusieurs étudiants, et être donné par un ou plusieurs professeurs.

Exemple:

Le cours de programmation objet en C++ est identifié par la clé unique: IFT1166. Ce cours est donné uniquement par M. Mohamed Lokbani qui est identifié par le code entier: 290, et 40 étudiants sont inscris à ce cours parmi eux : Marc LeCiel dont le code étudiant est l'entier: 98760. L'étudiant Marc LeCiel est inscrit aussi dans le cours de structures discrètes en informatique, identifié par la clé unique: IFT1063, et donné par Messieurs Bernard Gendron (code: 015) et Jean-Yves Potvin (code: 120) et 200 personnes sont inscrites à ce cours.

Vous devez écrire ce qui suit (Respectez le nom de chaque classe):

- la classe **Personne** (fichiers personne.h et personne.cpp).
- la classe **Etudiant** (fichiers etudiant.h et etudiant.cpp).
- la classe **Professeur** (fichiers professeur.h, professeur.cpp).

- la classe **Cours** (fichiers cours.h, cours.cpp).

vous devez représenter uniquement les deux relations suivantes:

un cours peut-être pris par un ou plusieurs étudiants.
un cours peut-être donné par un ou plusieurs professeurs.

Pour décrire les deux relations, vous devez agir uniquement sur la classe Cours et utilisez la classe **List** définie en 2.1. N'utilisez que les pointeurs (pas de tableau fixe), et l'utilisation de la librairie standard est interdite (pas de STL).

Des suggestions ….

Parmi les fonctions membres à écrire:

Constructeurs vide et avec paramètres ; le destructeur ; le constructeur de recopie ; les opérateurs d'affectation (=), de test d'égalité (==), l'opérateur de sortie (<<).

Pour le format de sortie: il est libre.

3. Tester vos programmes

Un fichier source main.cpp vous est fourni (dont une partie a été déjà décrite dans la planche de la démonstration #7). Une partie permet de tester votre liste générique, l'autre partie permet de tester l'implantation de l'héritage simple du problème mentionné en 2.2.

3.1 Test de la liste

Pour tester les outils de la liste uniquement:

g++ -Wall -DCLISTE -o Tliste main.cpp

Puis exécutez:

./Tliste

Le résultat obtenu (en tenant compte de mon format d'affichage) est en annexe 1.

3.2 Test de l'héritage

Compilez en premier chaque fichier à part (l'opition -c sert à réaliser cela):

g++ -Wall -c personne.cpp

g++ -Wall -c etudiant.cpp

g++ -Wall -c professeur.cpp

g++ -Wall -c cours.cpp

Compilez par la suite l'ensemble des fichiers avec le programme de test main.cpp

g++ -Wall -DCHERITE -o Therite personne.cpp etudiant.cpp professeur.cpp cours.cpp main.cpp

Puis exécutez:

./Therite

Le résultat obtenu (en tenant compte de mon format d'affichage) est en annexe 2.

4. Fichiers à remettre

Vous devez remettre 9 fichiers, respectez les noms définis pour les fichiers (y compris pour les majuscules et minuscules)

liste.h (va contenir la classe List, nœud et les fonctions membres)

personne.h (pour la classe Personne)

personne.cpp (pour les fonctions membres de la classe Personne)

etudiant.h (pour la classe Etudiant)

etudiant.cpp (pour les fonctions membres de la classe Etudiant)

professeur.h (pour la classe Professeur)

professeur.cpp (pour les fonctions membres de la classe Professeur)

cours.h (pour la classe Cours)

cours.cpp (pour les fonctions membres de la classe Cours)

Vous devez faire deux remises :

- Papier (à glisser sous la porte de mon bureau, Pavillon André-Aisenstadt, local 2249)
bien identifier sur le listing le cours et vos noms.

- Électronique : dans le répertoire où est votre programme, tapez:

remise ift1166 tp3 liste.h personne.cpp personne.h etudiant.h etudiant.cpp professeur.h professeur.cpp cours.h cours.cpp

5. Date limite

Mercredi 08 Décembre 12h00 (Midi).

6. Barème

Le tp3 est noté sur 15 points.

Les points seront répartis comme suit :

Exactitude : 7 points

un programme avec des avertissements "warnings" mais qui fait des choses non prévues dans la spécification 0/7

un programme qui ne compile pas 0/7

un programme avec des avertissements ("warnings") et qui donne les résultats escomptés 5.5

Modularité : 6 points

structure de votre programme, décomposition entre .h et .cpp, décomposition en fonctions, la manière avec laquelle vous allez coder votre programme etc.

Style : 2 point
le code doit être lisible, bien indenté et commenté.

Annexe 1

Sortie du programme Tliste (voir section 3.1):

```
------------------------------------------------------------
List(x) = 1.1 2.2 3.3 4.4
Removal of 1.1 in list x
Removal of 2.2 in list x
Removal of 3.3 in list x
Removal of 4.4 in list x
List(x) = liste vide
List(y) =1.1 2.2 3.3 4.4
List(z) =1.1 2.2 3.3 4.4
List(w) =1.1 2.2 3.3 4.4
le cas de tail
List(v) (tail of y) = 2.2 3.3 4.4
List(y)           = 1.1 2.2 3.3 4.4
List(i) = 1 2 3
List(ph) = C++ quel beau langage
List(ph) = C++ quel langage
List(ph) = Assurement C++ quel langage complique
Search(C++,ph) = 1
Search(belle,ph) = 0
List(ph) = C++ quel langage
List(ph) = quel
List(ph) = liste vide
Search(tout,ph) = 0
Search(belle,ph) = 0
liste[0]=tout
liste[1]=ne
List(liste) = tout ne marche pas toujours comme voulu !
List(copie) = tout ne marche pas toujours comme voulu !
List(liste) = liste vide
List(copie) = tout ne marche pas toujours comme voulu !
```

Annexe 2

Sortie du programme Tliste (voir section 3.2):

```
------------------------------------------------------------
1
1
1
0
Cours id: IFT1063

etudiant:
MIKE, JIM 1500
 LeCiel, Marc 98760

prof:
Potvin, Jean-Yves 120
 Gendron, Bernard 15

 Cours id: IFT1166

etudiant:
Jimy, mark 2000
 LeCiel, Marc 98760

prof:
Lokbani, Mohamed 290
```