INF1010

Programmation Orientée-Objet

Travail pratique #2
Constructeurs de copie, surcharge d'opérateur et agrégation et pointeurs intelligents

---

À remettre avant le mardi 4 février 2025, 23:30

À la fin de ce laboratoire, vous devrez être capable de :

- Comprendre le concept de constructeur de copie.
- Faire de la surcharge d'opérateurs.
- Utiliser des pointeurs intelligents.

Directives :

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Ne touchez pas aux entêtes des fichiers fournis sauf si on vous le demande.
- Les fonctions que vous décidez d'ajouter au programme doivent être documentées.

Conseils :

- Lisez les conseils, l'aperçu et les spécifications avant de commencer!
- Ayez lu vos notes de cours!
- Si vous avez des difficultés au cours du TP, rappelez-vous que de nombreux problèmes demandés sont assez couramment rencontrés en C++.
- Consulter la documentation sur cppreference, les notes de cours, et les forums sur Google peuvent vous donner de bonnes pistes de résolution!
- Si votre programme ne compile pas, veuillez mettre en commentaires les instructions qui ne compilent pas.
- Relisez votre code après l'avoir écrit pour éviter les erreurs de syntaxe.

Spécifications générales

- Tout warning à la compilation sera pénalisé. Si vous utilisez Visual Studio de Microsoft, vous devez activer l'option /W4. Sur Visual Studio, assurez-vous de compiler avec x64, certains warnings pourraient ne pas s'afficher sinon.
- Toutes les méthodes doivent être définies dans le fichier d'implémentation (.cpp) dans le même ordre que leur déclaration dans le fichier d'en-tête (.h). Le non-respect de cette règle entraînera une pénalité au niveau du style.
- Utilisez le plus possible la liste d'initialisation des constructeurs. L'utilisation du corps des constructeurs à la place de la liste d'initialisation entraînera une pénalité de style.
- L'ordre des variables (attributs) dans la liste d'initialisation doit être la même que celle dans la liste des attributs de la classe dans le fichier d'en-tête
- Suivez le guide de codage sur Moodle.

Mise en contexte

En tant que développeur stagiaire dans un hôpital, votre tâche sera de mettre en place un système de gestion du personnel. Dans un premier temps, vous serez amené à gérer les listes de médecins et d'infirmiers ainsi que leur salaire. Il est à noter que chaque médecin a une spécialité ou non.

Le projet contient les classes suivantes :
Medecin : Représente un médecin ayant une spécialité ou aucune.
Infirmier : Représente un infirmier.
Specialité : représente une spécialité d'un médecin
ReseauHopital : Un réseau d'hôpitaux qui conserve une liste d'hôpitaux.
HopitalPoly : Représente un hôpital qui conserve les listes de médecins et d'infirmiers.

Travail à réaliser

Les fichiers Medecin.h, Infirmier.h Specialite.h et HopitalPoly.h, ReseauHopital.h vous sont fournis, vous devez compléter les implémentations des classes.

Classe *Infirmier*

Cette classe est caractérisée par un nom, un prénom et un infirmier s'occupe d'une liste de chambres.

Cette classe contient les attributs privés suivants :

- Un nom (string)
- Un prénom (string)
- Un taux horaire (float)
- Un tableau de chambres (vector de string)
- Le nombre d'heures travaillées.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par copie qui copie tous les attributs de la classe.
- La surcharge de l'operator << qui permet d'afficher les attributs de la classe.
- La surcharge de l'operator + qui ajoute une chambre dans la liste des chambres si elle n'existe pas.
- La surcharge de l'operator - qui retire une chambre de la liste des chambres et incrémente le total des chambres.
- La surcharge de l'operator == qui permet de comparer deux infirmiers.

Classe Specialite

Cette classe est caractérisée par un domaine et un niveau.

Cette classe contient les attributs privés suivants :

- Un domaine (string)
- Un niveau (entier)

Les méthodes suivantes doivent être implémentées :

- Un constructeur par copie qui copie tous les attributs de la classe
- La surcharge de l'operator << qui permet d'afficher les attributs de la classe
- La surcharge de l'operator == qui permet de comparer deux spécialités.

Classe *Medecin*

Cette classe est caractérisée par un nom, un salaire, et une spécialité.

Cette classe contient les attributs privés suivants :

- Un nom (string).
- Une spécialité (Specialite).
- Un salaire.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par copie qui copie tous les attributs de la classe.
- La surcharge de l'operator << qui permet d'afficher les attributs de la classe.
- La surcharge de l'operator == qui permet de comparer deux médecins.

Classe *HopitalPoly*

Cette classe sert à sauvegarder les pointeurs de type Medecin et Infirmier.

Cette classe contient les attributs privés suivants :

- Le nom de l'hôpital (string).
- Un vector de pointeurs partagés à des médecins, qui contiendra les différents médecins.
- Un vector de pointeurs partagés à des infirmiers, qui contiendra les différents infirmiers.

Les méthodes suivantes doivent être implémentées :

- Un constructeur par copie qui copie tous les attributs de la classe.
- La surcharge de l'operator << qui permet d'afficher les attributs de la classe.
- La surcharge de l'operator += qui ajoute un médecin dans la liste des médecins s' il n'existe pas.
- La surcharge de l'operator += qui ajoute un infirmier dans la liste des infirmiers s'il n'existe pas.

- La surcharge de l'operator += qui fusionne deux hôpitaux en ajoutant les médecins et les infirmiers de l'un dans l'autre s'ils n'existent pas.
- La surcharge de l'operator == qui permet de comparer deux hôpitaux selon leur nom seulement.
- La surcharge de l'operator -= qui retire un infirmier dans la liste des infirmiers.
- La surcharge de l'operator -= qui retire un médecin dans la liste des médecins.
- La surcharge de l'operator -= qui retire un médecin selon son nom dans la liste des médecins

Classe *ReseauHopital*

Cette classe sert à conserver des hôpitaux.

Cette classe contient les attributs privés suivants :

- nom_, le nom du réseau d'hôpitaux.
- tableauHopitaux_ , un vector de pointeurs uniques d'hôpitaux.

Les méthodes suivantes doivent être implémentées :

- La méthode chercherHopital qui permet de chercher un hôpital dans le vecteur tableauHopitaux_
- La méthode afficher qui permet d'afficher les attributs de la classe.
- La surcharge de l'operator << qui permet d'afficher les attributs de la classe.
- La surcharge de l'operator += qui ajoute un hôpital dans la liste des hôpitaux s'il n'existe pas.
- La surcharge de l'operator -= qui retire un hôpital dans la liste des hôpitaux s'il existe.

Main.cpp

Les tests unitaires pourront être exécutés quand vous aurez des implémentations pour les classes. Pour les exécuter, vous devrez compiler le projet et exécuter le fichier main.cpp.

Correction

Veuillez remettre tous les fichiers .cpp SAUF le main.cpp dans un fichier .zip sous format Matricule1_Matricule2_TP2.zip

Également, SVP ne mettez pas de commentaires dans le code du fichier main.cpp

La correction du TP2 se fera sur 20 points.

Voici les détails de la correction :

- (4 points) Compilation du programme ;
- (4 points) Exécution du programme ;
- (2 points) Qualité du code;
- (10 points) Comportement exact de l'application