L3 ISFA – Programmation
Projet en binôme – Base de données d'une université

Date de rendu à préciser

Résumé

L'objectif de ce projet est de mener à bien un projet comportant des classes, en utilisant les notions d'héritage, de pointeurs et de résolution dynamique. Les exercices réutilisent tous des notions vues au cours des TP.

Notez bien que la division en exercices est artificielle : le projet peut se lire comme un seul long exercice.

Le projet s'effectue en binôme, aussi devrez-vous être prudents pour collectiviser le travail sans qu'il y ait de redondance.

Vous pouvez nous adresser vos questions (de préférence par mail) si vous avez des problèmes.

Consignes de rendu Vous devez constituer des binômes. Définissez vos binômes sur `https://docs.google.com/spreadsheets/d/1SJ9glw7Lw5EiZTdzz5pSRdNzY7gLElr-01yg8oAhVw8/edit#gid=0`.

La date de rendu sera précisée plus tard. Commencez cependant le projet rapidement, car aucun projet rendu en retard ne sera corrigé. Telle est la dure loi de la vie.

Votre rendu devra inclure tous les fichiers `.h` et `.cpp` que vous avez créé, la base de données que vous aurez créée, *et aucun autre fichier* (en particulier les `.cbp`).

Le code devra être soigneusement commenté afin d'expliquer vos choix de structure et de programmation.

En haut du fichier `main.cpp`, faites une zone de commentaires où vous rappellerez les noms de votre binôme.

L'objet de votre mail de rendu devra être : [Rendu Projet L3]. □

Modalités d'évaluation

Votre projet sera évalué sur les points suivants :

— Bon fonctionnement dans des conditions normales ;
— Robustesse (bon fonctionnement dans les cas extrêmes) ;
— Qualité et lisibilité du code source ;
— Qualité des tests ;
— Bonne compréhension et utilisation des concepts vus pendant le semestre.

Assurez-vous principalement que votre projet compile et fonctionne sans erreur d'exécution.

Si le code que vous rendez ne compile pas, vous serez notés sur 10. □

Exercice 1 : Création du projet

Le but de ce projet est de créer une interface (simplifiée) de gestion d'étudiants dans une université. Vous devrez créer des classes que vous réutiliserez et combinerez par la suite.

Commencez par créer un projet, et un fichier `main.cpp`, qui vous permettra de tester votre implémentation. Vous le compléterez au fur et à mesure des exercices.

Vous êtes en binôme pour un projet de programmation objet : vous pouvez donc vous diviser les tâches et travailler presque indépendamment. Commencez donc par vous répartir les différentes tâches à effectuer. **En haut de chaque fichier `.cpp`, précisez quel membre du binôme a rédigé la majeure partie du code.** Créez les fichiers `.h` et écrivez ensemble les prototypes des fonctions, puis effectuez chacun vos tâches.

Exercice 2 : Les classes

Les classes dont vous aurez besoin sont les suivantes :

1. Une classe `Etudiant`, que vous dériverez en deux classes filles `EtudiantLicence` et `EtudiantMaster` ;
2. Une classe `Cours`, que vous dériverez en deux classes filles `CoursLicence` et `CoursMaster` ;
3. Une classe `Enseignant` ;
4. Une classe `Universite`.

La classe `Etudiant` aura les attributs suivants :

1. Un numéro, en `unsigned int` ;
2. Un nom, en `std::string` ;
3. Une liste des cours pris par l'étudiant, et de leurs notes dans ces cours. Pour les représenter, vous créerez une structure de données `cours_avec_note` qui contiendra un `Cours*` et un `double` (qui représentera leur note dans le cours). La liste des cours pourra être représentée, au choix, sous la forme d'un `cours_avec_note*`, d'un tableau dynamique adapté du TD5 pour contenir des `cours_avec_note`, ou d'un conteneur de la STL (comme un `std::vector<>`) contenant des `cours_avec_note`.
4. Les `EtudiantMaster` pourront avoir une liste d'`EtudiantLicence` dont ils sont les tuteurs. Encore une fois, choisissez le moyen de les représenter qui vous semble le plus naturel.

La classe `Cours` aura les attributs suivants :

1. Un code, en `unsigned int` ;
2. Un nom, en `std::string` ;
3. L'enseignant responsable du cours ;
4. Une liste des étudiants qui suivent le cours, et leurs notes dans ce cours. Pour les représenter, vous créerez une structure `eleve` qui contiendra un `Etudiant*` et un `double`.

La classe `Enseignant` aura les attributs suivants :

1. Un numéro, en `unsigned int` ;
2. Un nom, en `std::string` ;
3. Une liste des `Cours` dont il est responsable.

La classe `Universite` contiendra une liste de tous les étudiants, une liste de tous les cours et une liste de tous les enseignants.

Créez les accesseurs pour chacune des classes.

Exercice 3 : Stockage des données

Afin que les données soient persistantes, vous allez les stocker dans des fichiers. Chaque `Etudiant`, `Cours` et `Enseignant` aura un fichier qui contiendra toutes les informations nécessaires pour recréer l'objet à chaque lancement du programme.

Créez, pour chaque classe, un constructeur prenant en argument un nom de fichier. Créez également une méthode permettant de modifier son propre fichier de données. **Pour cela, donnez à vos fichiers un nom pertinent.**

La classe `Universite` servira à coordonner toutes ces données. Créez un fichier de données contenant chaque nom de fichier pertinent, et une méthode permettant d'instancier tous les objets `Etudiant`, `Cours` et `Enseignant` de votre base de données en appelant les constructeurs adaptés. À partir de maintenant, au début de votre `main.cpp`, vous utiliserez cette méthode pour charger votre base de données et travailler dessus.

Exercice 4 : Fonctions d'`Etudiant`

Créez des méthodes permettant de déterminer la moyenne d'un `Etudiant`, ainsi que son classement dans chaque cours. Créez aussi une méthode permettant à un `EtudiantMaster` de savoir les `EtudiantLicence` dont il est tuteur qu'il lui faut aider.

Exercice 5 : Fonctions de `Cours`

Créez des méthodes permettant de calculer la moyenne sur un `Cours` et d'afficher le classement des étudiants qui la prennent, ainsi que d'identifier les étudiants en difficulté.

Exercice 6 : Fonctions d'`Enseignant`

Créez des méthodes permettant à un `Enseignant` de connaître les étudiants en difficulté dans les cours dont il est responsable.

Exercice 7 : Tests

Créez une base de données de taille conséquente (au moins cent étudiants et dix cours), et testez toutes les fonctionnalités. Vous aurez sans doute besoin d'ajouter des méthodes permettant d'afficher certains renseignements.

Afin d'effectuer les tests sans afficher des milliers d'informations, créez une boucle `do {} while ();` qui offrira à l'utilisateur des choix (comme afficher les étudiants en difficulté dans un cours en particulier, ou les étudiants en difficulté d'un enseignant spécifique, modifier les notes d'un étudiant, etc). Les choix seront gérés par un `switch` adapté. Le programme gérera ensuite tout via des méthodes de la classe `Universite`.