Compte rendu

Travaux pratiques de C++

TP 1

Sommaire

1. Présentation du TP

Le TP 1 consiste à créer un code permettant la gestion d'une bibliothèque. Nous devons gérer plusieurs aspects d'une bibliothèque, à savoir les livres disponibles, les auteurs présents, les lecteurs ainsi que leurs différents emprunts. Nous avons choisi de commencer par ce TP de difficulté débutant car nous sommes tous les deux débutants en Programmation Orientée Objet. Cela va donc nous permettre de continuer à nous familiariser avec les différentes notions.

1.1 Répartition et organisation du travail

Pour la création des différentes classes, nous nous sommes réparti le travail afin que chacun sache quelles classes créer. Pour la suite des questions, nous avons commencé en nous répartissant les sous questions, mais nous nous sommes rendu compte que nous devions travailler sur les mêmes fichiers. Nous avons donc changer de stratégie, Florent a fini le TP pendant que Colin commençait le TP suivant, afin de gagner du temps.

1.2 Outils utilisés

Nous avons utilisé l'outils de versioning GitHub afin de faciliter le travail en binôme. Cet outils nous a été très utile pour récupérer le code écrit par l'autre, notamment pour la création des classes. Comme éditeur de texte, nous avons utilisé Visual Studio Code. Cet éditeur est pratique car il nous permet de compiler et exécuter notre code directement depuis un terminal intégré à l'éditeur. Il nous a également permis d'exécuter les commandes nécessaires au versioning depuis le même terminal. Enfin, nous nous sommes servi de Copilot lors de la réalisation de tâches répétitives comme les getters.

2. Présentation du code

2.1 Organisation des classes

Nous avons créé une classe bibliothèque ainsi qu'une classe pour chaque partie de la bibliothèque que nous voulons gérer. Par exemple, pour la gestion des livres disponibles dans la bibliothèque, une classe Livre est créée. De même pour les auteurs, emprunts et les lecteurs. Une classe peut être utile pour d'autres classes, pas uniquement pour la Bibliothèque, comme par exemple la classe Livre qui contient des objets de type Auteur. De plus, une classe date est créée et est utilisée à chaque fois qu'une date est nécessaire (pour un emprunt par exemple).

2.2 Fonctions et méthodes

Au sein de la classe Bibliothèque, nous avons différentes méthodes permettant la gestion. Les auteurs, livres, lecteurs et emprunts sont stockés dans des vecteurs, il est donc nécessaire d'avoir des méthodes pour ajouter et enlever des éléments à ces vecteurs. L'ajout et la suppression d'un emprunt est bien sûr conditionné. La classe contient également des méthodes d'affichage, facilitant l'affichage de tous les livres par exemple, et faisant appel à des surcharges de l'opérateur <<. Afin de vérifier la disponibilité d'un livre nous avons préféré ajouter une fonction isDispo, qui scrute les emprunts en cours, et renvoie un booléen, 1 si dispo 0 sinon. Cette façon de faire nous a paru plus logique, c'est pour cela que nous avons voulu adapter cette partie, même si elle nous oblige à stocker les emprunts dans la classe bibliothèque.

2.3 Difficultés rencontrées

La principale difficulté que nous avons rencontrée est pour l'affichage d'un lecteur. Nous n'avons pas sur comment afficher la liste des livres empruntés par le lecteur dans la surcharge. La problème venait du for, nous n'avons pas su comment mettre tous les ISBN à la suite dans l'affichage. Pour contourner le problème, nous n'affichons pas les livres empruntés dans la surcharge. Nous avons créé une fonction dans le main, qui utilise la surcharge pour afficher les informations du lecteur. Ensuite, nous récupérons la liste des livres empruntés par le lecteur, puis les affichons à la suite. Cette solution n'est pas forcément optimale, mais fonctionne.

Une autre difficulté a été de trouver l'algorithme permettant de trier les lecteurs en fonction du nombre de livres empruntés. Pour cela, nous récupérons la liste des lecteurs de la bibliothèque afin de ne pas la modifier. Ensuite, tant que le nombre de livres empruntés par un lecteur est inférieur à celui de l'indice d'après, nous inversons les deux lecteurs afin d'avoir un tri par ordre décroissant. Nous pouvons enfin passer à l'affichage de la liste. Par la suite, son implémentation n'a pas posé de problèmes.

3. Conclusion

Ce TP nous aura permis de continuer notre familiarisation avec la POO en C++. Nous avons pu utiliser des notions vues en cours comme les surcharges d'opérateurs que nous n'avions jamais utilisées avant. Il nous a également apporté un gros travail sur les vecteurs, afin de bien comprendre leur utilisation et les différentes méthodes que nous pouvons utiliser.

Bien entendu, nous avons découvert le versioning avec GitHub, que nous pensions bien utiliser avant ce TP, et nous avons appris à nous en servir de la bonne manière.

Pour approfondir encore plus nos connaissances, nous aurions pu utiliser des pointeurs ou passages par références, ce qui sera fait au TP 3.