ESIREM
ÉCOLE SUPÉRIEURE D'INGÉNIEURS
NUMÉRIQUE ET MATÉRIAUX

Programmation C++
Compte Rendu TP n°1

Martinez Roméo
Septier Aubin

27 janvier 2023

Dans ce TP, nous avons abordé la programmation orientée objets (POO) en C++ en créant différentes classes pour gérer une bibliothèque.

Rapport créé avec la template latex ESIREM.

Table des matières

1 Introduction

Dans ce premier TP, l'objectif était de créer une application de gestion d'une bibliothèque en C++ (énoncé disponible sur : GitHub).
Nous avons choisi de commencer par ce TP afin de nous familiariser avec les bases du C++ et la maîtrise des classes, méthodes et autres fonctionnalités de ce langage orienté objets.
L'ensemble de notre projet est disponible et accessible sur GitHub).
Pour réaliser ce projet, nous avons tous les deux utilisé l'environnement de développement intégré Visual Studio Code et son extension C/C++. Nous avons utilisé la plateforme en ligne GitHub afin d'accéder au travail et au code de chacun facilement et efficacement. De plus, nous avons compilé notre code à l'aide d'un Makefile toujours dans cette idée d'efficacité.
A chaque séance, nous avons fait la même organisation : nous nous sommes réparti le travail à faire pour le début de séance, puis nous programmions, testions et réalisons des commit avant de mettre notre travail sur GitHub. Puis nous récupérions les avancées réalisées par l'autre avant de commencer une nouvelle tâche.

Comme dit précédemment, afin d'être efficaces et organisés, nous nous sommes réparti les différentes parties du projet. Pendant que l'un de nous s'occuper de programmer les bases des classes *Date*, *Reader* et *Borrow*, l'autre se chargeait des classes *Book* et *Author*. Ensuite, nous avons retravaillé les différentes classes afin de les rendre compatibles entre elles. Par exemple, au départ, l'auteur était une simple chaîne de caractères qui a été converti en objet avec la création de la classe Author. Pour cela nous avons dû arranger nos autres classes pour qu'elles prennent en compte que *Author* n'était plus une chaîne de caractères mais un objet.

2 Création des classes

2.1 Classe *Date*

La classe Date permet de gérer les dates, elle possède trois attributs privés qui sont _month, _day et _year qui représentent respectivement le mois, le jour et l'année de la date. Elle possède également des méthodes publiques qui permettent d'accéder à ces attributs, de les mettre à jour et de gérer les dates. En dehors de la classe Date, il y a des fonctions telles que isDate qui vérifie si la date passée en paramètre est valide, getDaysInMonth qui renvoie le nombre de jours dans le mois passé en paramètre, dayOfYear qui renvoie le jour de l'année pour la date passée en paramètre et toString qui renvoie la date sous forme de chaîne de caractères.

2.2 Classe *Book*

Nous avons créé la classe *Book* qui permet de créer des livres contenant les informations principales de cet objet comme le nom, l'auteur, la langue, etc. Nous avons ajouté des getters afin de pouvoir récupérer ces différentes informations et les réutiliser par la suite comme pour savoir si le livre est emprunté ou non par exemple. Cette classe réutilise deux autres classes créées qui sont la classe *Author* et *Date*. L'ensemble des informations peut être affiché grâce à une surcharge d'opérateur avec un entête fait en caractères ASCII pour rendre cela plus joli.

2.3 Classe *Reader*

Dans ce TP, nous avons décidé de créer la classe *Reader* pour gérer les informations des lecteurs de la bibliothèque. Cette classe contient des informations telles que le nom, l'adresse, le numéro d'identification des lecteurs. Nous avons utilisé des getters pour récupérer ces informations et des setters pour les mettre à jour. Nous avons également utilisé la classe *Date* pour stocker la date d'inscription des lecteurs. Nous avons implémenté une surcharge d'opérateur pour afficher les informations des lecteurs. Identiquement à la classe *Book* nous avons décidé d'embellir l'affichage graphique en utilisant des caractères ASCII. De plus la classe Reader est déclarée avec des constructeurs par défaut pour initialiser les valeurs des attributs. L'accès aux membres privés de la classe est géré par les méthodes publiques qui permettent d'obtenir ou de mettre à jour les informations du lecteur.

2.4 Classe *Author*

La classe *Author* permet d'ajouter un auteur dans notre programme qui sera identifié par un numéro d'identifiant pour le retrouver facilement, ainsi que par des informations essentiels comme le prénom, le nom et la date de naissance. La classe possède des getters afin de récupérer les paramètres de l'auteur et de setters pour mettre à jour les informations. Comme pour les classes *Book* et *Reader*, toutes les informations de l'auteur peuvent être affichées à l'aide d'une fonction utilisant une surcharge d'opérateur. À la base, l'auteur était une simple chaîne de caractères de la classe *Book* que nous avons dû transformer

en classe à part entière. Comme dans le cas des deux classes présentées précédemment, l'affichage des informations de l'auteur se fait grâce à des caractères ASCII formant un titre.

2.5 Classe *Borrow*

La classe Borrow est une classe qui permet de gérer les emprunts de livres dans une bibliothèque. Elle contient des informations sur le lecteur qui emprunte le livre, le livre en question et la date d'emprunt. Elle possède des méthodes pour ajouter un lecteur, ajouter un livre, rendre un livre, afficher les informations sur l'emprunt et vérifier si un livre est actuellement emprunté. Elle utilise des vecteurs pour stocker les informations sur les livres empruntés, tels que les identifiants des lecteurs et les ISBN des livres. La classe utilise également les classes Reader et Book pour stocker les informations sur le lecteur et le livre. Elle utilise aussi la classe Date pour stocker la date d'emprunt.

3 Création de la bibliothèque

3.1 Présentation de la classe

Finalement, nous avons implémenté la dernière classe de notre projet avec la création de la classe *Biblio*. Cette dernière permet de simuler une bibliothèque. Elle possède plusieurs méthodes permettant d'ajouter des livres, des auteurs et des lecteurs à la bibliothèque mais également de visualiser la liste des livres et auteurs la composant. Elle peut aussi indiquer si un livre est emprunté ou pas et afficher les informations de cet emprunt. Enfin, deux dernières méthodes permettent de rechercher les livres d'un auteur et de les afficher.

Nous n'avons pas pu implémenter les dernières méthodes demandées dans l'énoncé pour plusieurs raisons dont notamment le fait que le système permettant d'emprunter un livre dans la classe *Borrow* n'était pas du tout codé pour ça et qu'il aurait fallu réécrire une grande partie du code pour pouvoir les implémenter correctement.

3.2 Fonctionnement de la classe

Dans notre bibliothèque nous ajoutons les livres, auteurs et lecteurs avec leurs informations telles que les dates qui sont déclarées au préalable. Les livres, auteurs et lecteurs sont rangés dans des vecteurs afin de les stocker et pouvoir les récupérer simplement. Ces vecteurs sont alors réutilisés dans les méthodes d'affichage des livres, auteurs et lecteurs de la bibliothèque ainsi que pour la recherche des livres d'un auteur. Nous avons implémenté un système permettant d'emprunter des livres à l'aide de la classe *Borrow*. Ce système utilise également un vecteur permettant de savoir si le livre est déjà emprunté ou non et donc s'il est possible de l'emprunter. Nous pouvons également savoir, en pourcentage, combien de livres sont empruntés par rapport au nombre total de livres contenus dans la bibliothèque.

4 Conclusion

Au cours de ce TP, nous avons développé une application de gestion de bibliothèque en utilisant les concepts de programmation orientée objet en C++. Nous avons créé des classes pour gérer les livres, les auteurs, les lecteurs et les emprunts, en utilisant des méthodes, des getters, des setters et des surcharges d'opérateurs pour implémenter les fonctionnalités de base de la bibliothèque.

En cours de route, nous avons rencontré des difficultés, comme la nécessité de retravailler une partie de notre code pour pouvoir ajouter de nouvelles fonctionnalités ou pour adapter notre code à des classes supplémentaires. Nous avons également appris à travailler efficacement en équipe en utilisant des outils tels que Visual Studio Code, GitHub et un Makefile.

Au final, nous avons réussi à atteindre nos objectifs pour ce TP en créant une bibliothèque fonctionnelle. Ce projet nous a permis de renforcer nos connaissances en programmation C++ et de mettre en pratique nos compétences en programmation orientée objet. Cependant, nous avons encore des choses à apprendre pour notre futur métier d'ingénieur, notamment en ce qui concerne la programmation avancée en C++ et la mise en place de fonctionnalités plus complexes.