COMPTE RENDU TP INFORMATIQUE N° 1

Présentation du problème:

Dans ce TP nous avons dû réaliser un programme qui consiste à simuler une bibliothèque.
Dans un premier temps nous avons réalisé différentes classes notamment une classe Lecteur, une classe Livre, une classe Auteur, une classe Date, une classe Emprunt, et une classe Bibliothèque.

Nous avons choisi ce TP car l'un d'entre nous 2 débute en programmation, et nous voulions qu'il puissent programmer de la manière la plus autonome possible ainsi nous avons commencé par un TP de niveau 1 pour voir comment pour si ca allait ou non. Ensuite nous avons choisi au hasard l'un des 2 TP de niveau 1 sans vraiment de raison particulière.

Nous nous sommes réparti le travail de la manière suivante un de nous 2 c'est occupé de la création des classes de bases (Livre,Lecteur,Auteur,Date,Emprunt) qui sont nécessaire à la suite et l'autre c'est occupé de la création de la classe bibliothèque ainsi que de toutes les méthodes avancé lié à celle-ci.

Lors de ce TP nous avons travaillé de la manière suivante pendant que l'un de nous faisait le début du TP n°1 l'autre commençait le TP n°4.

Pour nous organiser dans notre travail nous avons utilisé github pour le versionning et visual studio pour l'IDE principalement car nous l'avions déjà installé avant les TPs et qu'il possède une interface github intégrer.

Dans ce TP la création de la plupart des classes nous est imposée dans l'énoncé, la seule dont la création ne nous est pas imposée est la classe Bibliothèque. Les objets de cette dernière sont composés d'un conteneur de Livre, un d'Emprunt, un de Lecteur, et un d'Auteur. Pour ces différents conteneur on a utilisé des std::vector car ils sont simples à utiliser et gèrent eux-même leurs allocation de mémoire.

```
class Bibliotheque
{
public:
    Bibliotheque();
    Bibliotheque(std::vector<Livre> Liste_Livre, std::vector<Auteur> Liste_Auteur, std::vector<Lecteur> Liste_Lecteur, std::vector<Emprunt
    std::vector<Livre> getListe_Livre();
    std::vector<Auteur> getListe_Auteur();
    std::vector<Lecteur> getListe_Lecteur();
    std::vector<Emprunt> getListe_Emprunt();
    int addLivre(Livre book);
    int addAuteur(Auteur writer);
    int addLecteur(Lecteur reader);
    int Emprunter(Livre* book, Lecteur* reader,Date* date);
    int Retour(Livre* book, Lecteur* reader, Date* date);
    std::vector<Livre> LivreparAuteur(Auteur author);
    float PourcentageLivreEmprunte();
    std::vector<Livre> LivreEmprunteParLecteur(Lecteur reader);

private:
    std::vector<Livre> _Liste_Livre;
    std::vector<Auteur> _Liste_Auteur;
    std::vector<Lecteur> _Liste_Lecteur;
    std::vector<Emprunt> _Liste_Emprunt;
};
```

[1] Classe Bibliothèque

Comme vous pouvez le voir dans l'image [1] la classe Bibliothèque possède aussi de nombreuses méthodes les importantes sont addLivre, addAuteur, addLecteur, Emprunter et Retour, mais les 3 premières utilisent la fonction pushback des vecteurs. La fonction Emprunter va quand à elle dans un premier temps regarder si le livre est disponible avec la méthode isdispo si le livre est bien disponible alors la fonction va utiliser la methode setdispo pour inverser la valeur de disponibilite puis elle va créer un objet Emprunt et va mettre cet objet dans _Liste_Emprunt enfin elle va ajouter l'ID du Lecteur dans la liste des personnes ayant emprunté le livre et elle va

aussi ajouter l'ISBN du livre dans la liste des livre ayant été emprunté par ce Lecteur.
La méthode Retour quant à elle utilise une boucle for parcourir _Liste_Emprunt on compare ensuite chaque élément du vector avec les objets passés en paramètre on considère qui si l'ISBN, l'ID du Lecteur ainsi que la Date d'emprunt sont les mêmes alors la personne peut bien rendre le Livre. On utilise ensuite un iterator pour pouvoir utiliser la fonction erase(on a pas utiliser les iterator pour la boucle car si _Liste_Emprunt a une taille de 1 lors de l'exécution de erase sa taille devient 0 et ducoup lors de prochain passage dans la boucle for il y a une erreur), enfin on inverse la valeur de disponibilité. A noter que ces 5 fonctions renvoient des int nous avons fait ce choix dans le cas ou l'on devrait un jour améliorer, on pourrait utiliser le renvoie d'entier pour signifier différentes choses pour l'instant 1 veut dire réussite et 0 veut dire échec. Nous avons aussi utilisé des try catch dans l'optique de rendre notre code plus robuste.

Nous n'avons pas rencontré beaucoup de problèmes, le plus gros que nous avons eu été un problème de compatibilité entre nos versions de visual studio. Nous avons aussi rencontré un problème lors de la fonction Emprunter plusieurs personnes pouvait un même Livre c'était dû au fait que l'on utilisait une copie de l'objet nous avons utilisé une référence à la place.

Ce TP nous a permis de revoir et de renforcer nos bases de C++ notamment sur la création de classe mais aussi sur l'utilisation de référence et de pointeur ainsi que sur l'utilisation de std::vector et des iterator.