# Compte rendu TP Programmation C++

## TP1 : bibliothèque et TP3 : magasin

**Etudiante du TD3 3A**

GAGNEPAIN Juliette

ALLARD Elsa

**Lors de ces 2 TP, nous avons travaillé avec les outils suivants :**

- **Visual Studio Code comme éditeur de texte et compilateur (via le terminal de type PowerShell)**
- **GitHub comme dépôt pour nos projets**

# I/ TP1 : Bibliothèque

## 1) Introduction

Ce TP concerne la gestion d'une bibliothèque. Cette bibliothèque comporte des livres qui peuvent être empruntés par des lecteurs en respectant la règle suivante : « Un Livre peut être emprunté par un seul Lecteur à un instant donné et doit donc être rendu à la bibliothèque par ce Lecteur avant d'être à nouveau emprunté par le même lecteur ou par un autre lecteur ».

Ce TP permet d'aborder la programmation orientée objets (POO) en accordant un certain nombre de classes simples qui vont interagir entre elles. Ce TP est donc adapté aux étudiants débutants la POO en C++.

Nous avons choisi ce TP de premier niveau car nous voulions nous évaluer sur un TP considéré comme « simple ». Dee plus n'ayant pas le même niveau en programmation, ce TP a permis de comprendre quelques notions de base pour l'une et approfondir son savoir pour l'autre.

Elsa s'est occupée de créer les fichiers .h des classes, et Juliette a implémenté les méthodes de ces dernières dans les fichiers .cpp et les fonctions permettant d'ajouter des fonctionnalités aux classes.

## 2) Résumé de notre travail

Dans un premier temps nous avons réalisé la création de différentes classes (auteur, livre, date, bibliothèque, emprunt, lecteur) nous avons implémenté les méthodes, getteurs et setteurs pour chacune d'entre elles. Voici l'exemple ci-dessous de la classe Livre :

```
12          public:
13                  //Constructor
14                  Book(string title, Author author, string language, string category, Date publication_date, int isbn);
15                  //Getteurs
16                  string title();
17                  Author author();
18                  string language();
19                  string category();
20                  Date publication_date();
21                  bool dispo();
22                  int isbn();
23                  //Setteurs
24                  void updateTitle(string title);
25                  void updateAuthor(Author author);
26                  void updateLanguage(string language);
27                  void updateCategory(string category);
28                  void updatePublicationDate(Date publication_date);
29                  void updateIsbn(int isbn);
30                  void updateDispo(bool dispo);
31                  //Surcharge de l'opérateur <<
32                  friend ostream& operator<<(ostream& os, Book& book);
33
34          private:
35                  string _title;
36                  Author _author;
37                  string _language;
38                  string _category;
39                  Date _publication_date;
40                  int _isbn;
41                  bool _dispo = true;
42  };
43
```

Pour finir cette première étape, nous devions nous assurer que cette classe soit fonctionnelle. Pour ce faire nous avons alors réalisé un programme test (ici test de la classe Livre. Nous avons donc refait ces étapes pour la totalité de nos classes créer.

```
57      // //Book class test
58      // book l("A brief history of time", authors), "English", "Popularization", Date(), 0001);
59      // authorsa =  l.author();
60      // a.updateLastName("Hawking");
61      // cout << a.last_name() + "\n";
62      // l.updateLanguage("French");
63      // cout << l.language() << "\n";
```

Dans un second temps, nous avons implémenté les méthodes/fonctions helper demandées. Puis nous avons surchargé l'opérateur << (sortie standard sur la console) pour différentes classes (ci-dessous « livre.h ») afin d'afficher les informations d'une instance de cette classe pour l'utilisateur.

```
ostream& operator<<(ostream& os, Book &book)
{
    os << book.title() << " (isbn : " << book.isbn() << "), written by " << book.author() << " in " << book.language() << " on " << book.publication_date() << ", category "
        if(book.dispo() == true)
        {
                os << "available";
        }
        else
        {
                os << "unavailable";
        }
    return os;
}


#endif
```

Enfin, pour nous permettre de se servir de nos classes nous avons créé dans le programme principal une bibliothèque avec plusieurs livres, auteurs et lecteurs. L'utilisateur est libre d'écrire dans cette partie du projet les actions qu'il veut faire (emprunter un livre, …).

```
    Library lib = Library();

    Author rowling = Author("J.K.", "Rowling", 0001, Date(31, 07, 1965));
    Author camus = Author("Albert", "Camus", 0002, Date(07, 11, 1913));
    Author hugo = Author("Victor", "Hugo", 0003, Date(26, 02, 1802));
    Author proust = Author("Marcel", "Proust", 0004, Date(10, 07, 1871));
    Author flaubert = Author("Gustave", "Flaubert", 0005, Date(12, 12, 1821));

    Book harryPotter = Book("Harry Potter à l'école des sorciers", rowling, "English", "Fantasy", Date(26, 06, 1997), 0);
    Book animauxFantastiques = Book("Les Animaux fantastiques", rowling, "English", "Fantasy", Date(01, 03, 2001), 1);
    Book etranger = Book("L'Etranger", camus, "French", "Novel", Date(01, 01, 1942), 2);
    Book peste = Book("La Peste", camus, "French", "Novel", Date(10, 06, 1947), 3);
    Book miserables = Book("Les Misérables", hugo, "French", "Novel", Date(01, 01, 1862), 4);
    Book condamne = Book("Le Dernier Jour d'un condamné", hugo, "French", "Novel", Date(01, 02, 1829), 5);
    Book tempsPerdu = Book("A la recherche du temps perdu", proust, "French", "Novel", Date(01, 01, 1913), 6);
    Book swann = Book("Du côté de chez Swann", proust, "French", "Novel", Date(14, 11, 1913), 7);
    Book bovary = Book("Madame Bovary", flaubert, "French", "Novel", Date(15, 12, 1856), 8);

    Reader elsa = Reader("elsaa", "Elsa", "Allard", {});
    Reader juliette = Reader("julietteg", "Juliette", "Gagnepain", {});

    cout << elsa;
    cout << hugo;
    cout << etranger;

    Loan loanElsa = Loan(Date(05, 01, 2022), 1, elsa.id());
    Loan loanJuliette = Loan(Date(07, 01, 2022), 1, juliette.id());
```

## 3) Conclusion

Elsa : Ce TP a été relativement simple sur les débuts et s'est complexifié sur la fin. J'ai pu comprendre comment fonctionnaient les classes en C++ et en écrire quelques-unes. Lors de ce TP Juliette m'a expliquée au mieux ce qu'elle faisait pour qu'ensuite j'essaye de le faire par moi-même.

Juliette : Ce TP m’a permis de me remémorer la syntaxe du langage C++, qui a pris un peu de temps. La partie la plus longue et redondante a été la déclaration et l’implémentation des getteurs et setteurs pour chaque classe.

## II / TP3 : Magasin

### 1) Introduction

Ce TP consiste à concevoir une application de magasin en ligne nommée EasyStore.

Ce TP permet d'aborder la programmation orientée objets (POO) en accordant un certain nombre de classes simples qui vont interagir entre elles. Ce TP est donc adapté aux étudiants ayant un niveau intermédiaire en C++.

Nous avons choisi ce TP de niveau « intermédiaire » car nous le trouvions plus intéressant et nous avons plus réussi à nous projeter dans le code à écrire en lisant le sujet.

Elsa s'est occupée de créer les fichiers .h des classes, et Juliette a implémenté les méthodes de ces dernières dans les fichiers .cpp et les fonctions permettant d'ajouter des fonctionnalités aux classes.

### 2) Résumé de notre travail

Comme dans le TP précédent, nous avons commencé par créer différentes classes (client, commande, produit et magasin) ainsi que leurs getteurs et setteurs. Exemple de classe Client :

```
12  class Client{
13
14          public:
15                  //Constructors
16          Client();
17                  Client(int id, string first_name, string last_name, vector<Product> purchases);
18                  //Guetters
19                  int id();
20                  string first_name();
21                  string last_name();
22                  vector<Product> purchases();
23                  //Setteurs
24                  void updateId(int id);
25                  void updateFirstName(string first_name);
26                  void updateLastName(string last_name);
27          void addPurchase(Product product);
28                  void deletePurchase(Product product);
29          void clearPurchases();
30                  void updateQuantityPurchase(Product product, int quantity);
31                  //Surcharge de l'opérateur <<
32                  friend ostream& operator<<(std::ostream& os, Client& client);
33
34          private:
35                  int _id;
36                  string _first_name;
37          string _last_name;
38          vector<Product> _purchases;
39  };
```

Pour ce TP nous avons également réaliser une surcharge d'opérateur dans nos classes comme ci-dessous pour la classe Client :

```
41  inline ostream& operator<<(ostream& os, Client& client)
42  {
43      os << client.first_name() << " " << client.last_name() << " (id : " << client.id() << ") : " << "   Purchases :\n";
44      for(int i = 0; i < client.purchases().size(); i++)
45          {
46                  os << client.purchases()[i].label() << " : " << client.purchases()[i].details() << " (" << client.purchases()[i].price() << " €\n";
47          }
48      return os;
49  }
50
51  #endif
```

Nous avons pu remarquer que ce TP n'était pas véritablement différent du TP précédent. Néanmoins les consignes donnant moins de détails sur les procédures à suivre.

Par la suite nous avons implémenté les fonctionnalités demandées pour les différentes classes. Cette partie a pris un peu de temps du a certaines erreurs qui persistent.

Enfin, nous avons commencé à créer à l'intérieur du programme principal un menu qui permet de sélectionner une action à réaliser puis un sous menu pour chacune des ces actions avec les options correspondantes pour la gestion du magasin et de ses composantes (les produits, les clients et les commandes).

```
//Menu
int choice = 0;

cout << "                                EasyStore --- Welcome !\n\n\n";
cout << "1 - Store\n";
cout << "2 - Clients\n";
cout << "3 - Orders\n\n";
cin >> choice;

//Gestion des produits du magasin
if(choice == 1)
{
    cout << "   1 - Display all products\n";
    cout << "   2 - Add a product\n";
    cout << "   3 - Search a product\n";
    cout << "   4 - Update quantity of a product\n";
    cout << "   5 - Back\n";
    cin >> choice;

    if(choice == 1)
    {
        store.productsDisplay();
    }
    else if(choice == 2)
    {
        int id;
        string label;
        string details;
```

## Conclusion

Elsa : Ce TP a été relativement simple sur les débuts et s'est complexifié sur la fin. J'ai pu comprendre comment fonctionnaient les classes en C++ et en écrire quelques-unes. Cependant j'ai encore des difficultés mais j'ai pu mettre en œuvre ce que Juliette m'avait expliqué lors du premier TP pour pouvoir intégrer les notions de bases.

Juliette : A part le menu de gestion du magasin, je n'ai pas eu l'impression d'une grande différence de difficulté entre le TP1 et le TP3. En lisant l'énoncé et avec ce que nous avions fait lors du TP1, j'arrivais à imaginer le code à écrire, et dans certains cas, cela ne différenciait pas tellement du premier TP en terme de difficulté.

## III / Conclusion générale

**Elsa** : Lors de ces deux TP, j'ai pu en apprendre plus sur les bases du C++. Je partais de zéro mais grâce à Juliette qui a su me guider et me donner des conseils pour réaliser les tâches que nous nous étions répartie j'ai pu mieux comprendre la programmation C++. Cependant, j'ai trouvé le TP1 plus simple que le TP3, car il y avait plus d'indications. J'ai pu aussi connaître les diverses fonctionnalités de Github. Pour finir sur cette conclusion je pense que le fait d'être accompagné lors de la programmation peut m'aider pour la réalisation de projets.

**Juliette** : J'ai trouvé ces 2 TP intéressants dans la mesure où je n'avait pas programmé en C++ depuis quelques mois. La structure étant la même pour les 2 TP, nous aurions pu prendre un sujet qui s'en éloigne. Durant ces TP, j'ai pu expérimenter le travail d'équipe (répartition des tâches, planification des séances de travail…), je me suis remémoré la syntaxe du langage C++ et certaines commandes pour ajouter des fichiers sur GitHub depuis un terminal Visual Studio Code. Je pense que le fait de travailler sur des projets plus longs sur la durée et plus difficiles me permettrait de mieux optimiser mon code afin que mon programme soit plus efficace.