Библиотека

Линк към хранилището в *Github:* https://github.com/KrisCvertanov/Library

1. Увод

Разглежданият проект реализира информационна система, която поддържа библиотека. Поддръжката е спрямо наличност на книгите, характеристики и достъп на потребители до системата, както и до съответните операции, които могат да бъдат извършени върху нея. Програмата съхранява и обработва данните във файл както за наличните книги, така и за потребителите в системата.

За реализирането на програмата трябва да бъдат постигнати следните цели и задачи:

- Подходящо представяне на книга за програмата, така че да се обхванат повечето й важни характеристики
- Подходящо представяне на потребителите, техният вид и достъп до системата
- Изграждане на структура, реализираща библиотека. Тя ще обхваща множество от книги, ще извършва промени по наличността им, различни операции за намиране, подреждане и предоставяне на информация по тях. Системата ще съдържа набор от потребители, като дадени лица ще могат да правят промени по съхраняваните данни за тях.
- Правените промени по библиотеката да могат да бъдат записвани във файлове, но да има и избор за отхвърлянето им.

Документацията за проекта е разделена на 3 слоя:

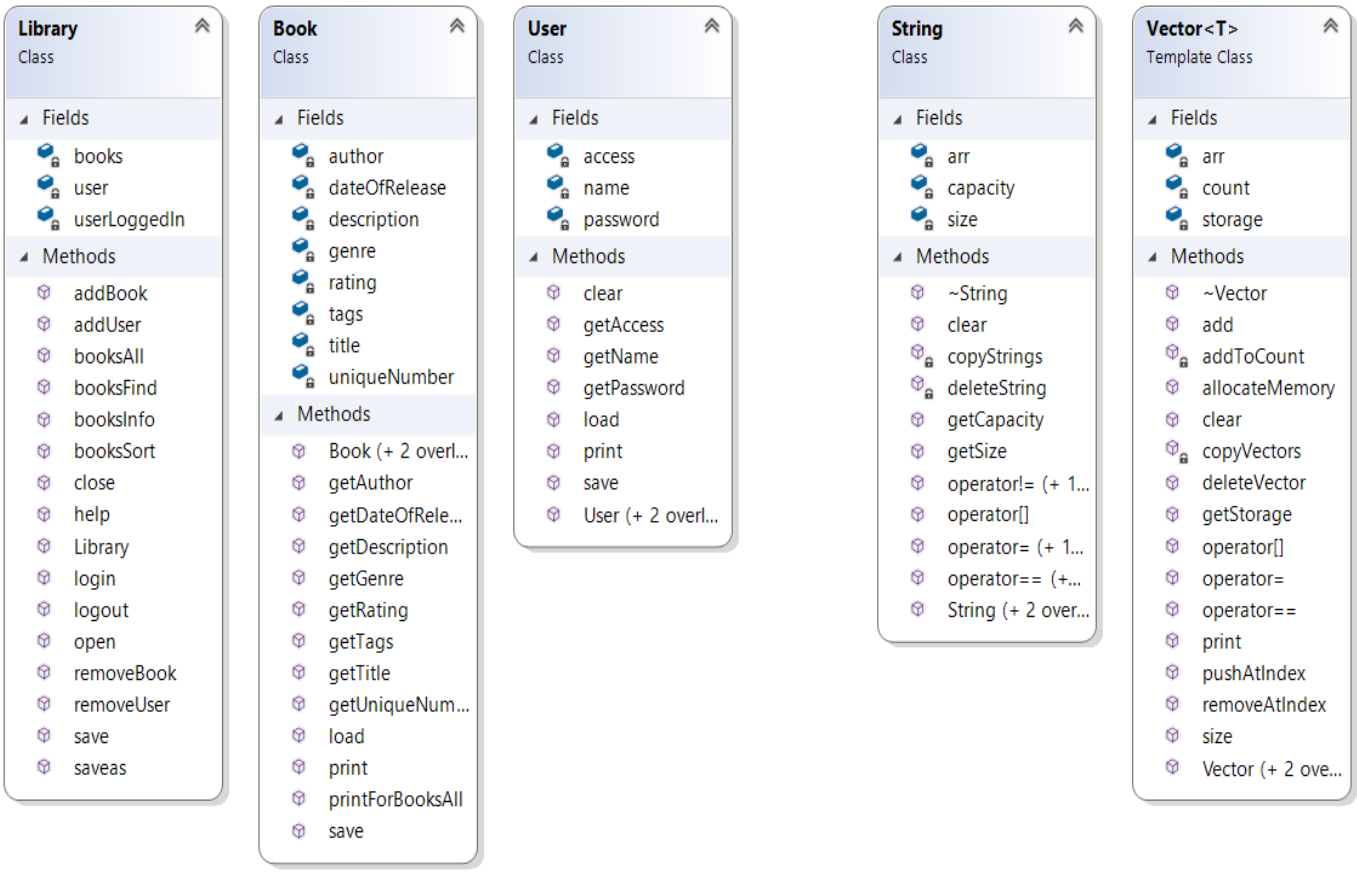
1. Увод
2. Проектиране
3. Реализация и уточнения

2. Проектиране

В проектирането на програмата се стреми да се следват добрите принципи и практики на ООП дизайна. Реализирани са 5 класа:

- Library
- Book
- User
- String
- Vector<T>

Те имат следните характеристики и йерархия:



Диаграма 2.1

1. **Клас String**: осъществява преоразмеряващ се динамичен масив от символи. Има следните три полета:
   - char* arr;
   - int size;
   - int capacity;

   Член-данната size следи за големината на масива, а capacity – за заделената памет за масива. Класа има следните методи:
   - String() - конструктор по подразбиране
   - String(const char*, cons tint, const int) - конструктор с параметри;
   - String(const String&) - копи-конструктор;
   - String& operator=(const String&) - оператор равно спрямо друг обект от тип String;
   - String& operator=(const char*) - оператор равно спрямо низ;
   - ~String() - деструктор;

- void copyStrings(const char*, const int, const int) - функция, достъпна само за класа, копираща информация по зададени параметри. Следва добрите принципи на ООП;
- void deleteString() - изтрива заделената памет за масива. Следва добрите принципи на ООП;
- friend std::istream& operator>>(std::istream&, String&) - оператор за вход. Чрез него се въвежда обект от тип String.
- friend std::ostream& operator<<(std::ostream&, String&) - оператор за изход. Чрез него извежда обект от тип String.
- friend std::ifstream& operator>>(std::ifstream&, String&) - оператор за четене от файлов поток.
- friend std::ofstream& operator<<(std::ofstream&, String&) - оператор за писане във файлов поток.
- char& operator[](const int) - оператор за достъп до даден елемент на обекта
- bool operator==(const String&) const - оператор за сравняване на два обекта от типа, казва дали са еднакви или не.
- bool operator==(const char*) const - оператор за сравняване на обект от тип String и низ, казва дали са еднакви или не.
- bool operator!=(const String&) const - оператор за различаване на два обекта от типа, казва дали са различни или не.
- bool operator!=(const char*) const - оператор за различаване на String и низ, казва дали са различни или не.
- int getCapacity() const - метод за достъп до заделената памет
- int getSize() const - метод за достъп до големината
- void clear() - изчиства заделената памет за масива

2. **Клас Vector**<T>: осъществява преоразмеряващ се динамичен масив от шаблонен тип. Има следните три полета, аналогични на клас String:
    - T* arr;
    - int storage;
    - int count;

    Методи:

    - Vector() - конструктор по подразбиране;
    - Vector(int) - коснтруктор с параметър int;
    - Vector(const Vector&) - копи – конструктор;
    - Vector& operator=(const Vector&) - оператор=;
    - ~Vector() - деструктор
    - void add(const T&) - добавя елемент към края на вектора.;
    - int getStorage() const - метод за достъп на заделената памет;
    - int size() const - метод за достъп на големината ;

- void print() const - извежда елементите на вектора;
- void pushAtIndex(const T&, int) - на зададен индекс се присвоява новата стойност;
- void removeAtIndex(int) - премахва на зададен индекс елемента. Големината на вектора намалява с 1;
- void clear() - изчиства заделената памет за масива;
- void deleteVector() - изтрива заделената памет за масива;
- void copyVectors(const Vector&) - копира информация от друг вектор;
- void addToCount() - увеличава големината с 1(достъпен само за класа!);
- void allocateMemory(int) - заделя подаденото количество памет за вектора
- bool operator==(const Vector<T>&) const - сравнява дали два вектора са еднакви
- T& operator[] (int) const - оператор за досъп до даден елемент

Класовете String и Vector<T> са съществени за проекта, понеже по – голямата част от полетата на останалите класове са от такива типове и спестяват работата с динамична памет в следващите класове:

3. **Клас Book**: представя книга за програмата. Има следните полета:
    - String author - автор
    - String title - заглавие
    - String genre - жанр
    - String description - описание на книгата
    - String dateOfRelease - дата на издаване
    - String tags - ключови думи
    - double rating - оценка на книгата
    - int uniqueNumber - сериен номер в библиотеката

    Методи:

    - Book() - конструктор по подразбиране
    - Book(const String&, const String&, const String&, const String&, const String&, const String&, double, int) - коснтруктор с параметри(String)
    - Book(const char*, const char*, const char*, const char*, const char*, const char*, double, int) - коснтруктор с параметри(char*)
    - const String& getAuthor() const - метод за достъп до автора
    - const String& getTitle() const - метод за достъп до заглавието
    - const String& getGenre() const - метод за достъп до жанра
    - const String& getDescription() const - метод за достъп до описанието
    - const String& getDateOfRelease() const - метод за достъп до датата на издаване

- const String& getTags() const - метод за достъп до ключовите думи
- const String& getRating() const - метод за достъп до оценката
- const String& getUniqueNumber() const - метод за достъп до серийния номер
- void print() const - извежда подробна информация за книгата
- void printForBooksAll() const - извежда подбрана информация за книгата
- friend std::ifstream& operator>>(std::ifstream, Book&) - оператор за четене от файлов поток
- friend std::ofstream& operator<<(std::ofstream, const Book&) - оператор за писане във файлов поток
- friend std::istream& operator>>(std::istream, Book&) - оператор за вход
- friend std::ostream& operator<<(std::ostream, const Book&) - оператор за изход
- void save(std::ofstream&) - метод за запазване на характеристиките на книга във файлов поток
- void load(std::ifstream&) - метод за извличане на характеристиките на книга от файлов поток

4. **Клас User**: представя потребител в системата. Полета на класа:
   - String name - име ;
   - String password - парола;
   - bool access - достъп(1 за администратор, 0 за нормален потребител).

   Методи:
   - User() - конструктор по подразбиране;
   - User(const String&, const String&, const bool) - конструктор с параметри(String);
   - User(const char*, const char*, const bool) - конструктор с параметри(char*);
   - const String& getName() const - метод за достъп до името;
   - const String& getPassword() const - метод за достъп до паролата;
   - bool getAccess() const - метод за достъп до нивото на достъп;
   - void print() const - извежда информацията за потребител;
   - void clear() - изчиства заделената памет за потребител;
   - friend std::ifstream() operator>>(std::ifstream, User&) - оператор за четене от файлов поток;
   - friend std::ofstream() operator<<(std::ofstream, const User&) - оператор за писане във файлов поток;
   - friend std::istream() operator>>(std::istream, User&) - оператор за вход;

- friend std::ostream() operator<<(std::ostream, const User&) - оператор за изход;
- void load(std::ifstream&) - метод за извличане на потребител от файлов поток.
- void save(std::ofstream&) - метод за записване на потребител във файлов поток;

5. **Клас Library:** осъществява представянето за библиотеката. Полета за класа:
    - Vector<Book> books - вектор от книгите за библиотеката;
    - User user - текущо влезлият потребител(ако има такъв);
    - bool userLoggedIn - казва дали има влязъл потребител в системата.

    Методи:

    - Library() - конструктор по подразбиране;
    - void open(const char*, Vector<Book>&, Vector<User>&) - отваря и зарежда информацията от файл(както за книгите, така и за потребителите). Ако не е отварян никакъв файл, може да се изпълняват само командите open, help и exit. **Не изисква** влязъл потребител;
    - void close(Vector<Book>&, Vector<User>&) - затваря текущия файл като изчиства промените по библиотеката. **Не изисква** влязъл потребител;
    - void save(const char*, Vector<Book>&, Vector<User>&) const - запазва направените промени в текущо отворения файл. **Не изисква** влязъл потребител;
    - void saveas(const char*, Vector<Book>&, Vector<User>&) const - запазва промените по библиотеката в нов файл. **Не изисква** влязъл потребител;
    - void help() const - извежда списък с поддържаните команди от системата. **Не изисква** влязъл потребител;
    - void login(const char*, Vector<User>&) - потребител влиза в системата(ако няма вече влязъл). **Не изисква** влязъл потребител;
    - void logout() - потребител излиза от системата(ако има влязъл). **Изисква** влязъл потребител;
    - void booksAll() const - извежда подбрана информация за наличните в библиотеката книги. **Изисква** влязъл потребител;
    - void booksInfo(int) const - извежда подробна информация за книга с търсения сериен номер. **Изисква** влязъл потребител;
    - void booksFind(const char*, const char*) const - намира и извежда подробна информация за книга. Търсенето е по име, автор или ключова дума. **Изисква** влязъл потребител;

- void booksSort(Vector<Book>&, const char*, const char* = “asc”) const - сортира вектора с книги по име, автор, година или оценка. Може възходящо или низходящо. **Изисква** влязъл потребител;
- void addUser(const char*, const char*, Vector<User>&) const - добавя потребител в системата по зададени име и парола(непосредствено след командата. Името и паролата не съдържат интервали!). **Изисква** влязъл администратор;
- void removeUser(Vector<User>&) const - премахва потребител от системата. След командата се дава нов ред и се въвежда име. **Изисква** влязъл администратор;
- void addBook(Vector<Book>&) const - добавя нова книга към библиотеката. След командата се дава нов ред и се въвеждат характеристиките на книгата. **Изисква** влязъл администратор;
- void removeBook(Vector<Book>&) const - премахва книга от библиотеката. След командата се дава нов ред и се въвежда серийният номер на книгата, която ще се премахва(серийният номер е единствената уникална характеристика на книга). **Изисква** влязъл администратор;

3. Реализация и уточнения

Класовете **String** и **Vector<T>** са единствените, в които има динамично заделяне на памет. Това значително подпомага и улеснява реализацията на останалите 3 класа. При всяко използване на оператора new се прави проверка дали динамичната памет е заделена успешно или не. Пример:

```
arr = new(std::nothrow) char[1];

if(arr == nullptr){
      throw (“No memory for string!”);

}
```

Въвеждането на обект от тип **String** става символ по символ.

Клас **Book**:

- датата на издаване е в посочения в условието формат
- ключовите думи са разделени с точно един интервал
- в условието, книгата има характеристика: година на издаване. При мое обърквaне, клас книга в библиотеката има характеристика: дата на издаване.

Така или иначе, програмата ползва само годината, а ден и месец - не. Няма загуба или промяна в исканата от условието функционалност свързана с характеристиката.

Клас **User**:

- името и паролата са с по една дума
- ако потребителят е администратор, то нивото му на достъп е 1. В противен случай е 0

Клас **Library**:

- има команди, които за да бъдат изпълнени, е нужен влязъл потребител(администратор). Тези подробности са отбелязани в точка 2.
- при изпълняване на команда **open** се търси файл с подаденото име. Ако такъв не съществува, се създава нов празен с това име. Ако отворен файл е празен, в него се записва символа '0', защото се очаква първото прочетено нещо да е цяло положително число(броят на съхранените книгите).
- Програмата съхранява данните за потребителите във файл, въпреки че това не е посочено в условието. Затова този файл е с фиксирано име: "users.txt". Никъде не се налага въвеждането на името на файла с потребители от ползващия програмата. Ако не съществува такъв, при команда **open** се създава нов празен с фиксираното име. При празен файл се записва потребителят по подразбиране посочен в условието – администраторът.
- Ако няма отворен файл или векторът с книги е празен, при опит за ползване на команда **close** се извежда съобщение за грешка.
- Понеже трябва да има опция да се отхвърлят направените промени по библиотеката и системата, а не да се извършват директно, в главната функция се създават временни вектори за потребители и книги. Всички направени промени се извършват върху тях, като при опция **save** или **saveas** те се записват в дадения файл. Тези промени са достъпни за библиотеката едва след повторно ползване на опцията **open**! Тогава на оригиналния вектор се присвоява временния.
- Методите, ползващи оригиналния вектор с книги за библиотеката са: **booksAll(), booksInfo(int), booksFind(const char*, const char*),** понеже не извършват промени.
- При командата **users remove** не е възможно да се изтрие администраторът от системата.

Общи подробности:

- Между всеки две думи в команда има **точно един** интервал.
- Всяка команда се въвежда по посочения в условието начин. Не може да започва или завършва с интервал.
- Приема се че в името на файловете няма интервали

- При **технически** или **логически** неправилно използване на командите се извежда съответното съобщение за грешка.
- Тестове има в хранилището в *Github*