**Library**

**Глава 1. Увод**

Главната идея на проекта е да изпълнява команди свързани с малка библиотека. Проекта има „база данни“ , което позволява добавянето на нови книги и потребители, запазвайки записите между различни потребители и сесии. На този етап приложението е във вид на конзолен команден интерпретатор.

Зададени изисквания към проекта:

* Всяка книга трябва да има уникален идентификатор
* Да се поддържат два вида потребители – администратор и потребител
* Да се спазват правилата за правата на съответните роли

Целта на проекта е да покрива следните основни команди:

* Отваряне и прочитане на съдържанието на файл по заден път до него
* Помощна функция, принтираща всички поддържани команди
* Запазване на файла и новото съдържание
* Запазване на новото съдържание в нов файл по зададен нов път
* Изчистване на текущо зареденият файл, но продължаване на програмата
* Изход от програмата
* Вписване като съществуващ потребител
* Излизане от текущият потребител
* Добавяне и премахване на книга
* Добавяне и премахване на потребител
* Сортиране на книгите по даден критерии
* Пълна информация по зададен идентификатор
* Кратка информация за всички книги в базата данни
* Намиране на книга по зададен критерии

**Глава 2. Преглед на предметната област**

Основни дефиниции, които ще бъдат използвани са:

* *Deserialization* – процес по превръщане на серия от байтове (низ) в обект.
* *Serialization –* процес по превръщане на обект в серия от байтове.

Използваният алгоритъм за сортиране на книгите е std::sort, който използва *IntroSort.* За търсенето по критерии е използване линейно търсене.

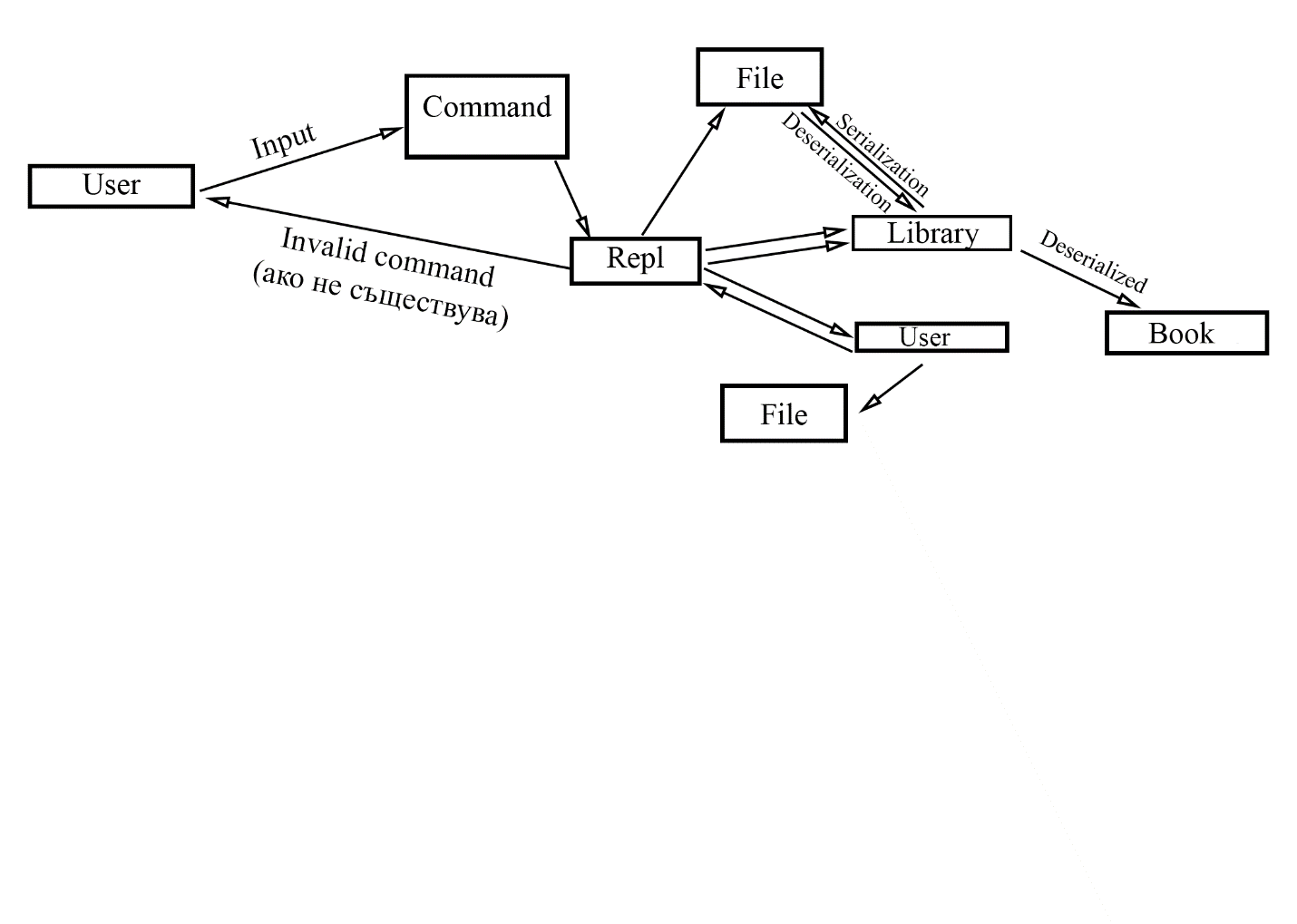
Първият проблем при изпълнението на заданието, е локализирането, отварянето и прочитането на файла, който съдържа базата данни. Необходимо е да не се нарушава неговата цялост и той да остане годен ползване, след приключването на програмата. При подаден невалиден адрес на файла, се създава нов празен файл.

Друг възможен проблем при изпълнението на програмата е, опит за изпълняване на операции преди успешно да бъде отворен файл, и съдържанието да бъде прочетено и *Deserialized*. В този случай потребителят се известява, че първо е необходимо да отвори файл и да се впише в системата.

Невалидно съдържание или оформление на файла, също може да доведе до редица проблеми и неточности по време на изпълнението на програмата. Методът Library::Deserialize() проверява дали, редовете във файла съдържащ базата данни е кратен на 8 (8 са полетата определящи всяка една книга, като всяко едно поле е на нов ред). Ако не е, програмата извежда съобщение и не изпълнява този метод. До този метод на представяне на информацията се достигна, след възникването на редица проблеми при представянето на всяка книга, като един единствен ред. При този подход нямаше начин, по който да се разграничат ясно отделните полета, ако в тяхната стойност имаше интервал.

Поради нуждата от запазване и работа с низове, като главен предмет на програмата, ще бъде използвано std::string за всички низове в програмата. А за множество от низове ще бъде използвано std::vector<std::string>.

**Глава 3. Проектиране**

*Примерна диаграма на последователностите*

Основните класове са:

* Repl
* File
* Library
* User
* Book

На графиката е представен основния процес и стандартното протичане на програмата. От потребителя се изисква въвеждането на команда, ако тя е валидна се изпълнява, ако не е се принтира подходящо съобщение на екрана и се очаква от него, нова команда.

За изпълнението на програмата е отговорен класът Repl и неговият метод Repl::REPL(). Класът има полета Library\*::library, File::file User::user, които осъществяват връзката между него и съдържанието на отвореният файл, чието съдържание е записано в file.content, а списъкът с обекти в library.Deserialized.

Library::Library (const vector<string> content) { this->Deserialized (); }

След отварянето и прочитането на файла, съдържанието му се подава на конструктора на класа **Library** и в него се извиква методът Deserialization(), който има за цел да преведе всеки един ред от съдържанието на файла, в нов обект от тип Book и да го добави към колекцията от книги на библиотеката.

За ползване на команди свързани с библиотеката, е нужно да потребителят да бъде вписан. Като за добавяне/премахване на книга или потребител, е задължително той да бъде администратор.

**Глава 4. Реализация, тестване**

При стартирането на програмата, се създава обект от клас Repl, който съдържа полета library и file, които преди отваряне на валиден файл, са с *default* стойности. След отваряне и прочитане на съдържанието на файла, методът Deserialization(), го подготвя за употреба и записва в полето library.books. Този метод е отговорен за извличането на информацията за всеки от елементите, като:

* Идентификатор
* Автор
* Заглавие
* Жанр
* Описание
* Година на издаване
* Ключови думи
* Рейтинг

Ако успешно бъде отворен файл, то променливата bool file\_Opened; променя своето състояние, което не позволява отново да се отвори друг файл, докато не се затвори текущият с командата Close.

За вписване е нужно потребителят да въведе командата *login,* след това последователно бива питан за име и парола. За въвеждането на паролата със скрити символи е ползвана отделна функция Hidden\_password().

**(към т.1 от „Бонуси“)** Се ползва функция \_getch(),от библиотеката: conio.h, защото тя позволя въвеждане на символи без „*echo*“ (а именно, въведеният символ от потребителят, да не се показва на конзолата). До нея се достигна след проучване в <https://stackoverflow.com/> и <https://www.youtube.com/>. В процеса на разработка се достигна до проблем при въвждането на ***Enter(\r\n)***, тъй като цъкълът while (c=\_getch()){}, прекратява своето изпълнение при засичане на символът ***\r,\t ,*** а в буфера остава ***\n,*** който трябва да бъде изчистен на следващ етап, за да не бъде добавен към следващата команда.

За таблицата *Users* от „базата данни“, се ползва текстов файл, в който всеки един запис се запазва във формат <потребителско име> ̺ <парола>, като всеки запис е на нов ред. За път до файла е зададен адресът на проектът (или папката на .exe файлът). /*Users.txt*/

**(към т.3 от „Бонуси“) „**Търсене на книга по зададен критерий да игнорира регистъра на буквите (малки или големи)“, се използва функцията:

transform(tokens[2].begin(), tokens[2].end(), tokens[2].begin(), ::tolower);

Тя прилага фунцкията ::tolower на всеки елемент от началото на низа tokens[2], до неговия кра, като при сравняването със записите от файла, също я прилага за всяко едно заглавие, автор или таг (спрямо избраният критерии).

Всяка въведена валидна команда с нужните аргументи, съответства на метод от класа Library или User. При невалиден идентификатор или друг аргумент, програмата извежда подходящо съобщение и продължава своето изпълнение.

За запазване на промените в текущият файл се използва командата save , а за запазване в нов файл – saveas. За принтиране на всички поддържани команди и нужните аргументи, е нужно използването на командата help. Бъдещо подобрение би могло да бъде динамично добавяне към колекцията с поддържаните функции.

Важни сценарии, които трябва да бъдат тествани са:

* Празен файл
* Несъществуващ файл
* Добавяне на книги от потребители, които не са администратор
* Добавяне на потребители от потребители, които не са администратор
* Принтиране на книга с невалиден идентификатор
* Ползване на команди свързани с библиотеката, от потребители, които не са вписани

**Глава 5. Заключение**

Проекта на този етап изпълнява всички поставени цели и функционалности от начален етап. Възможните подобрения за разширяването на функционалността на проекта, са много на брой, но на преден план биха били, подобрения свързани с добавянето на нови критерии за сортиране на книгите, или възможност за множество от критерии.

Източници:

<https://www.youtube.com/>

<https://stackoverflow.com/>

Дамян Иванов, ФН: **82057, спец. Компютърни науки, поток.1, гр.3**

<https://github.com/damianivanov/CS-SU-OOP/tree/master/Library>