**СОФИЙСКИ УНИВЕРСИТЕТ – “СВЕТИ КЛИМЕНТ ОХРИДСКИ” - СОФИЯ**

**КУРСОВА РАБОТА**

Тема: Изготвяне на приложение за управление на библиотека

Изготвена от: Александър Захариев Каратов, специалност

Информационни Системи, I курс

СОФИЯ

2020

**Увод**

Основната цел е изграждане на приложение, чрез което да може да се управлява набор от налични книги и потребители. Потребителите имат различно ниво на достъп в средата и някои заявки към програмата са им забранени. Задължителна е възможността за запазване във файлове (на книгите и на потребителите).

Задължителните операции свързани с файловете са: „open“, „save“, „save as“ и „close“. За всички файлове са текстови с разширение „ .txt “. Операцията „open“ трябва да отваря конкретен файл и зарежда в паметта неговото съдържание (open <file\_name>.txt). Операцията „save“ записва промените, изпълнени по вече заредения файл в него. Операцията „save as“ запазва изготвените промени по заредения в паметта файл в указан от потребителя файл (save as <file\_name>.txt). Командата „close“ трябва да затваря заредения в паметта файл.

Заявките, които могат да бъдат изпълнявани без да има потребител, влязъл в профила си са: „open“, „close“, „save“, „save as“, „login“, „help“ и „exit“. Всяка друга команда или заявка изисква наличието на активен потребител. Останалите команди, които трябва да се реализират са: „logout“, „books all“, „books info“, „books sort“, „books add“, „books remove“, „users add“ и „users remove“. Всички команди, свързани със създаване или изтриване на книги и потребители трябва да бъдат позволени само на потребители с административен статус на „администратор“.

**Първа глава -** **Преглед на предметната област**

* 1. **Основни дефиниции, концепции и алгоритми, които ще бъдат използвани**

Този проект е изграден чрез: управление на динамична памет, обектно програмиране, използване на файлове като вход и изход на програмата, алгоритъм за сортиране – “***Insertion Sort***”. Управлението на динамичната памет се изразява в заделяне на памет за нужните обекти и изчистването на паметта, когато данните за тези обекти са изтрити. Обектното програмиране определя създаването на структури от данни, наречени обекти, които имат изградена сигурност чрез методи, които се използват за въвеждане на данни в обекта и извеждане на данни от него за използване в алгоритми, за представяне на конзолата или записване във файл. Алгоритъмът за сортиране “***Insertion Sort***” се използва за представяне на въведените в програмата книги, сортирани във вид, избран от потребителя. Алгоритъмът се състои от три ключови елемента: 1. Итериране от първия до n-тия елемент на даден масив; 2. Сравнение на текущия елемент с елемента преди него; 3. Ако текущият елемент е по-малък от предходния се проверява дали е по-малък от елементите преди предходния елемент. По-големите елементи се преместват една позиция напред с всяко преместване, за да се освободи място за сменения елемент. Този алгоритъм отговаря на фигура 1.1.

*Фиг. 1.1 Пример за осъществяване на алгоритъма „****Insertion sort****“*

* 1. **Дефиниране на проблеми и сложност на поставената задача**

Основната задача е едновременно поддържане на две множества – на потребителски профили и книги в оперативната памет на компютъра. Съответно спрямо множеството от заредени потребители в оперативната памет се определя възможността за влизане на потребител в своя профил, както и заявките, които ще може да прави спрямо книгите и/или потребителите. Основната сложност идва от това, че трябва да бъде реализиран контейнер, подобен на масив, но с възможност за добавяне и изтриване на елементи – вектор. Трябва да бъдат осъществени и валидации, така че да не се позволява записването на невалидни данни във вектора и съответно във файл.

* 1. **Подходи и методи за решаване на поставените проблемите**

За изграждането на база от потребителски профили и книги се използват обекти – User и Book, както и шаблонния обект Vector. Един обект от тип User съхранява в себе си всички данни на един потребител, обектът Book аналогично за една книга, а шаблонният обект Vector има възможността да изгради списък от елементи от еднакъв тип данни. Съответно за осъществяване на операциите по създаване, изтриване, задаване на стойност на обект се осъществява така наречената „Голяма четворка“ – изграждане на конструктор (без параметри), копиращ конструктор, предефиниран оператор „=“ и деструктор. За улеснение при дефинирането на обекти е реализиран и конструктор с параметри, отговарящи на параметрите на обектите (важи за Book и User). За осъществяване на връзка между програмата и файловете е нужно предефиниране на оператор „>>“ и оператор „<<“ или изготвяне на методи за конкретните класове.

* 1. **Потребителски изисквания и качествени изисквания**

Да се напише програма, реализираща информационна система, която поддържа библиотека. Програмата съхранява и обработва данни за наличните в момента книги във файл.

Всяка книга се характеризира със  следните данни:

* автор
* заглавие
* жанр
* кратко описание
* година на издаване
* ключови думи
* рейтинг
* уникален номер за библиотеката

Системата поддържа два вида потребители — администратори и клиенти на библиотеката. Всеки потребител се характеризира със следните данни:

* потребителско име
* парола
* ниво на достъп — указва дали потребителят е администратор или не.

След като приложението отвори даден файл, то трябва да може да извършва посочените на фигура 1.2 операции, в допълнение на общите операции (open, close, save, save as, help и exit):

|  |  |
| --- | --- |
| login | След въвеждането на командата потребителят последователно е питан за потребителско име и парола. Ако потребител с посочените данни съществува в програмата, се извежда съобщение “Welcome, <username>!”, където <username> съответства на  потребителското име. В противен случай се извежда съобщение за грешно име или парола. При повторен опит за login, се изкарва съобщение “You are already logged in.” |
| logout | Потребителят напуска системата (програмата продължава да работи) |
| books all | извежда последователно за всяка книга следната информация:  - заглавие, автор, жанр,  персонален номер |
| books info <isbn\_value> | извежда на екрана подробна информация за книга с персонален номер равен на <isbn\_value>  Пример: books info 1124 |
| books find <option> <option\_string> | <option>е едно от title, author, tag  <option\_string>е стойността на критерия за търсене, може да съдържа интервали  Примери: books find title Introduction to programming  books find author Stephen King  books find tag superhero |
| books sort <option> [asc | desc] | <option>е едно от title, author, year, rating  asc означава възходящо сортиране (по подразбиране), а desc означава низходящо сортиране  Примери:  books sort title  books sort year desc |
| users add <user> <password> | Добавя нов потребител с потребителско име <user> и парола <password>. Потребителят и неговата парола се записват във файл. |
| users remove  <user> | Изтрива потребителя с потребителско име <user> от файла. |

*Фиг. 1.2 Задължителни за реализация команди*

При първоначално стартиране на програмата няма налични данни за книги. Има регистриран по подразбиране само един потребител с администраторски акаунт със следните данни:

* потребителско име: admin
* парола:   i<3c++

Програмата очаква да се въведе команда, като след въвеждането и се изпълнява според дефинираните правила. Това продължава до въвеждането на командата “exit”, която прекратява програмата.

Фигура 1.3 описва за всяка от командите дали е достъпна само при коректно влязъл потребител и дали е ограничена само за потребителя admin.

|  |  |  |
| --- | --- | --- |
| команда | изисква ли потребител? | само за администратор? |
| open | не | не |
| close | не | не |
| save | не | не |
| saveas | не | не |
| help | не | не |
| login | не | не |
| logout | да | не |
| exit | не | не |
| books all | да | не |
| books find | да | не |
| books sort | да | не |
| books view | да | не |
| books add | да | да |
| books remove | да | да |
| users add | да | да |
| users remove | да | да |

*Фиг. 1.3 Таблица с изисквания към нивото на достъп на потребителите, спрямо всяка команда*

 Бонус:

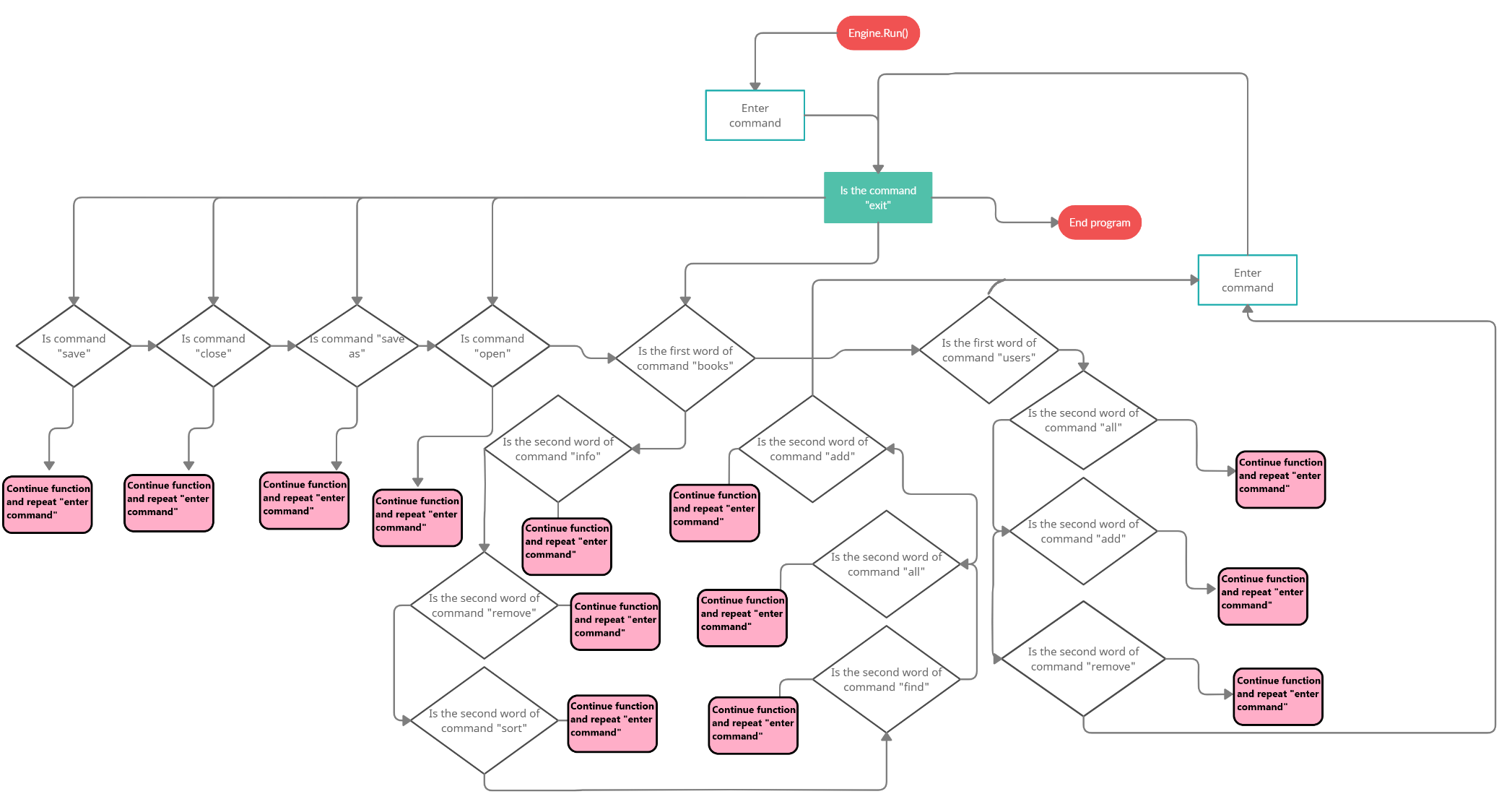
* при въвеждане на паролата на екрана да се изписва  символа \* вместо реалния символ
* при сортиране на книгите по зададен критерий, да се напише алгоритъм различен от пряка селекция и метода на мехурчето
* Търсене на книга по зададен критерий да игнорира регистъра на буквите (малки или големи)

**Втора глава – Проектиране**

* 1. **Архитектура – ООП дизайн**

Програмата е изградена от пет класа: Book, User, String, Vector и Engine. Обектът Book отговаря за създаването, валидацията и изтриването на една книга от оперативната памет. Обектът User аналогично отговаря за един потребител. Обектът String е дефиниран като низ от символи, използва се за да замени *char\**. Шаблонният обект Vector се използва за запазване на множество елементи от един тип. По този начин е възможно генерирането на „вектор“ от елементи (Book, User, String, и съответно всички типове данни в езика c++), като към него може да се добавят елементи или да се изтриват. Обектът Engine служи за премахване на уязвимостта при използване на „main“ файла за извършване на цялата логика по управление на програмата. Чрез дефиниране на обект Engine всички библиотеки, използвани в Engine, както и използваните скрити в Engine променливи не се дефинират в „main“ файла. В „main“ файла е включена само една библиотека – „Engine.h“, инстанциран е само един обект от тип Engine и е изпълнен само един метод – Engine.Run(). По този начин кодът, който е „двигателя“ (Engine) се използва, но „main“ файлът няма достъп до останалите класове.

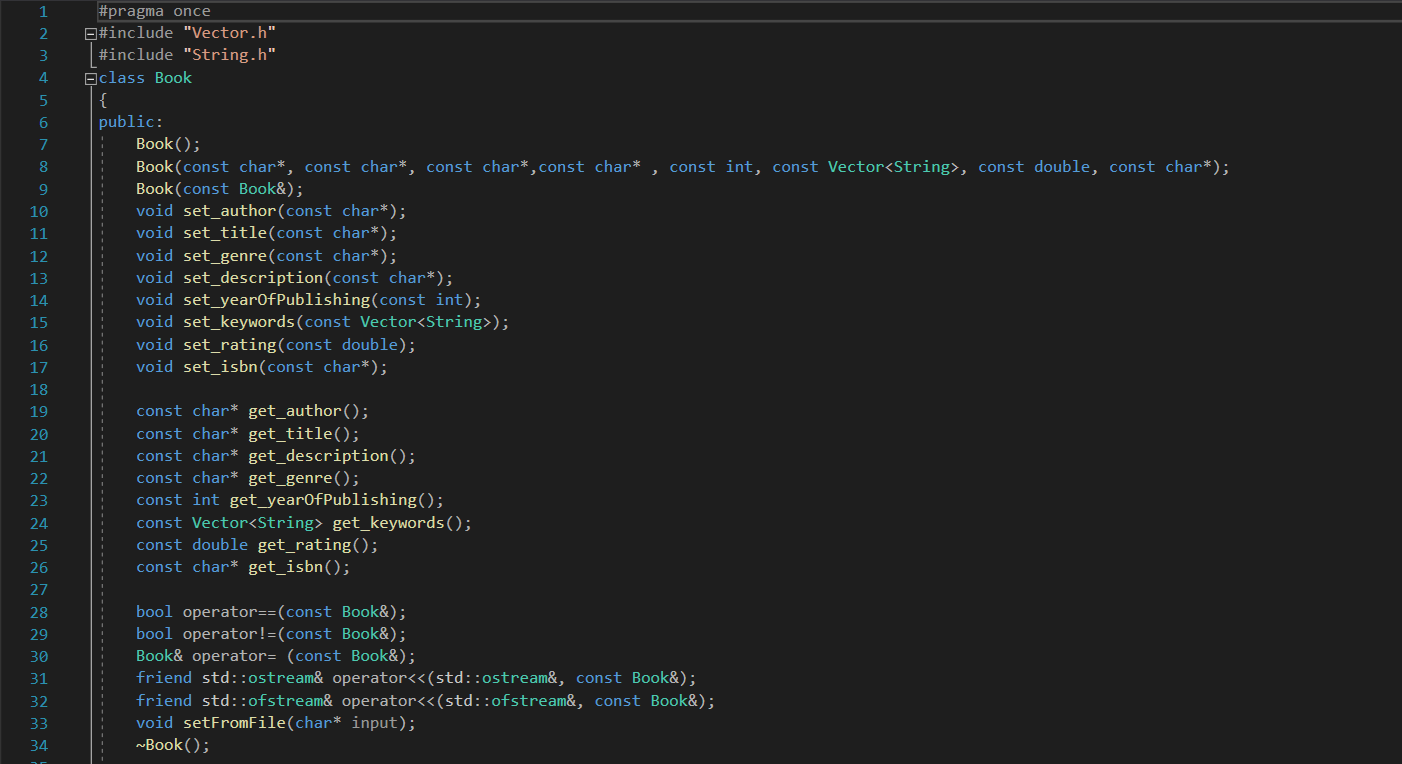
На фигура 2.1 е изобразена блок схема, показваща как програмата изпълнява своята функционалност. (Фигурата не включва валидациите за всяка функционалност.)



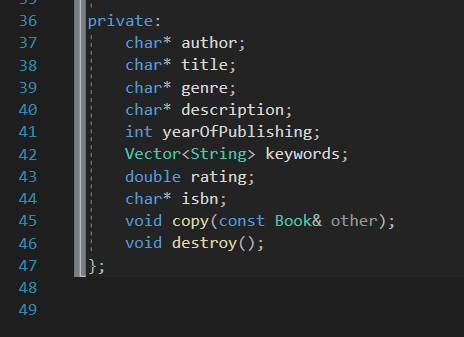
*Фиг. 2.1 Блок схема на програмата*

**Трета глава – Реализация и тестване**

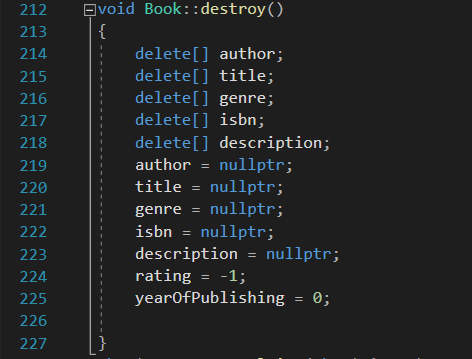
* 1. **Реализация на програмата**

На фигури 3.1 и 3.2 е представена структурата на клас Book. На фигура 3.1 са изобразени конструкторите, методите и предефинираните оператори за класа Book, които са публични. На фигура 3.2 са изобразени частните (не публичните) методи и член-данни на класа Book. Реализирано е динамично управление на паметта за съответните елементи (автор, заглавие, жанр, описание, ISBN номер) чрез деструктор. Показан е на фигура 3.3.

*Фиг. 3.1 Конструктори, методи и предефинирани оператори за класа Book*

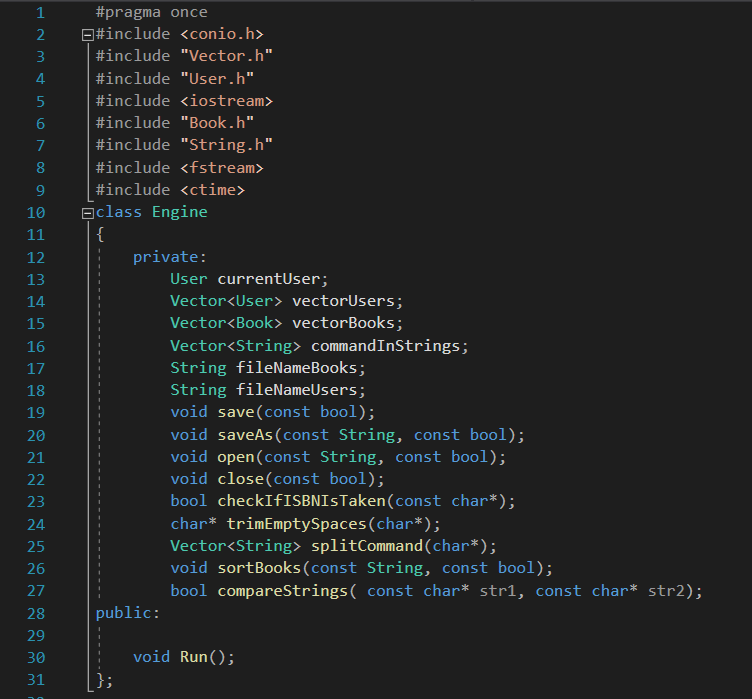


*Фиг. 3.2 Частни методи и член-данни на класа Book*

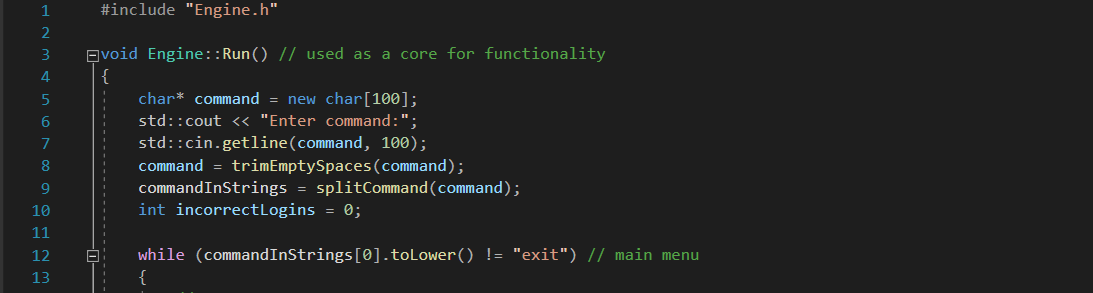
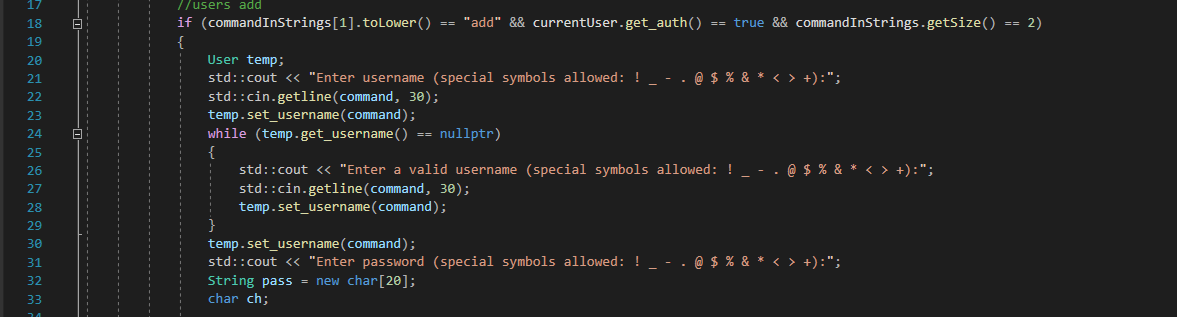
**

*Фиг. 3.3 Деструктор на класа Book*

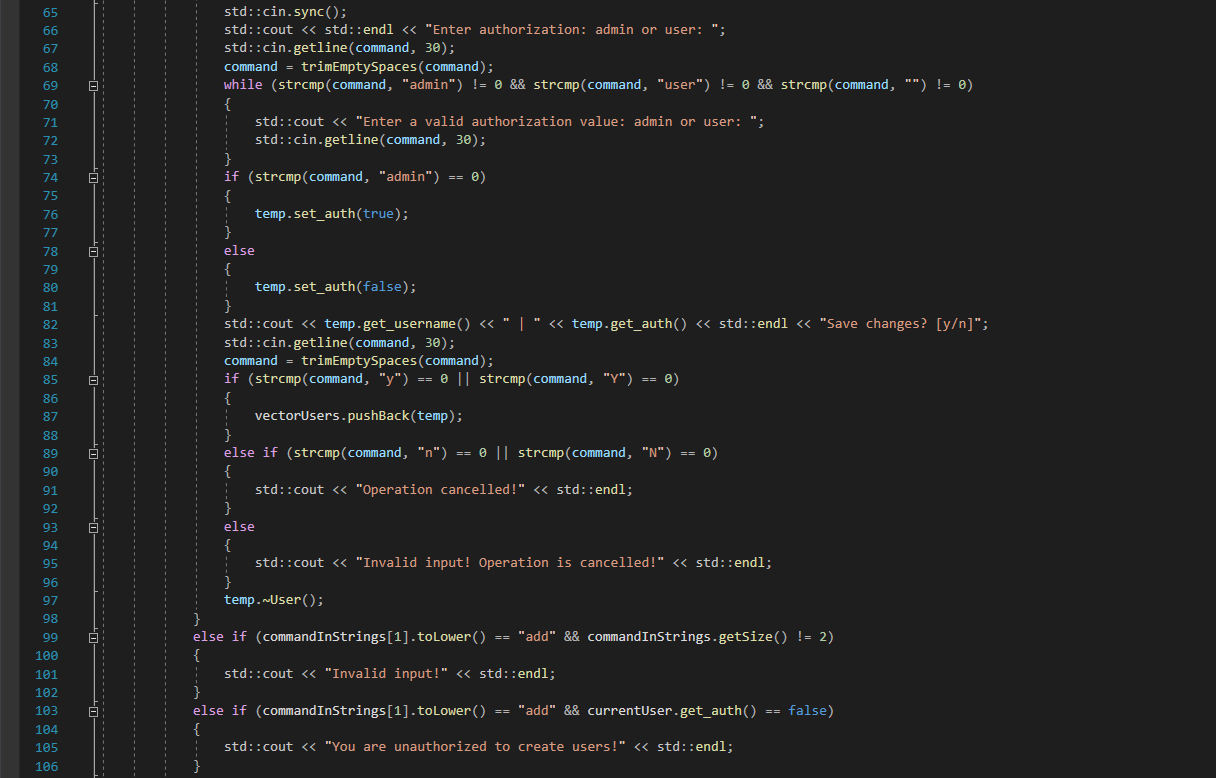
На фигура 3.4 е представена структурата на класа Engine. Наблюдава се единствен публичен метод Run(), частни параметри: currentUser (представлява активния в системата потребителски профил), vectorUsers (съдържа всички потребителски профили, прочетени от файл), vectorBooks (съдържа всички книги, прочетени от файл), commandInStrings (съдържа разделената на думи команда), fileNameBooks (представлява името на файла за книги), fileNameUsers (представлява името на файла за потребителски профили), както и частни методи: save (изпълнява запазване в отворения файл, в зависимост от избор за книги и потребители – изборът на вид файл важи за всички команди за файлове), saveAs (изпълнява запазване във файл по избор на потребителя), open (изпълнява отварянето на файл по избор на потребителя), close (затваря файл), checkIfISBNIsTaken (проверява векторът от книги за съвпадащ ISBN номер), trimEmptySpaces (изтрива празните пространства преди и след командата, не изтрива пространства между думи в командата), splitCommand (разделя командата на думи), sortBooks (използва се алгоритъмът **Insertion sort** за реализация на сортиране на книгите по параметри избрани от потребителя), compareStrings (метод за сравняване по лексикографски начин на два низа от символи). Програмата изпълнява цикъл с проверки на командите и изпълняването им, докато не се въведе команда „exit“. Осъществени са валидации относно: правото на достъп на потребителите до сензитивни команди, въвеждането на невалидни или грешни стойности и въвеждане на грешни, непълни или невалидни команди. На обикновените потребители е забранено да изпълняват командите: „books add“, „books remove“, „users add“, „users remove“, „users all, забранено е изпълнението на всякакви команди (освен за работа с файлове) при липсата на активен потребител. Изпълнена е валидация за пароли и потребителски имена при създаване на потребител – трябва потребителското име и паролата да се състоят от 8 или повече символа, задължително да съдържат главна и малка буква, цифра и специален знак. При изписване на парола на конзолата се появява знак „\*“. При създаването на книги е изпълнена валидация за дължина на заглавието – до 30 символа, името на автора – до 30 символа, жанра – до 20 символа, описанието – до 100 символа, както и проверка за дължината – точно 13 символа (задължително от 0 до 9) и повторение на ISBN номера, изпълнява се проверка и за стойността на оценката за всяка книга (тя трябва да е между 0 и 10), както и за годината на издаване на книгата (тя трябва да е преди текущата година), изпълнява се проверка за дължината на ключовите думи. Проверките се изпълняват в методите за задаване на конкретните параметри. На фигури 3.5 - 3.22 е представена имплементацията на класа Engine с реализираните методи и функционалности.



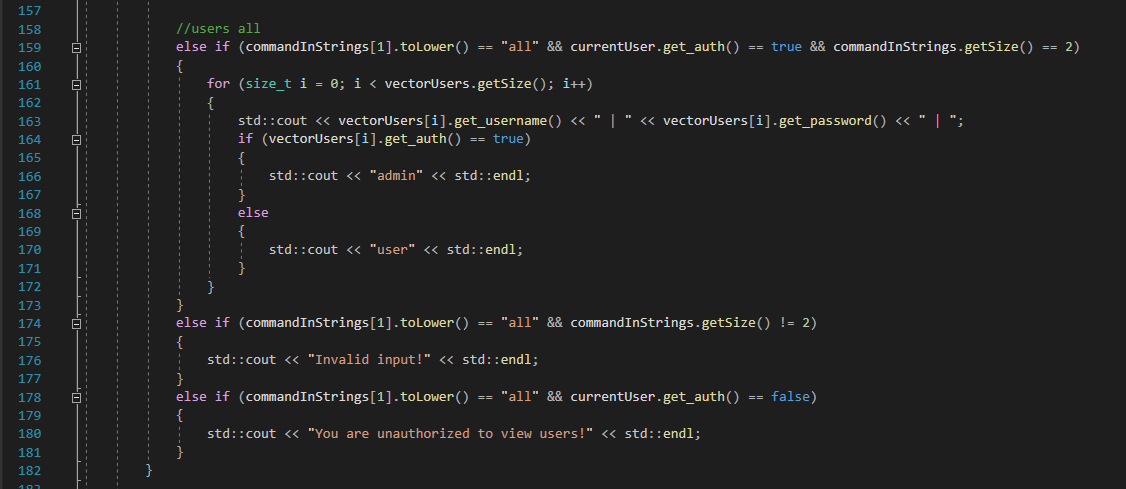
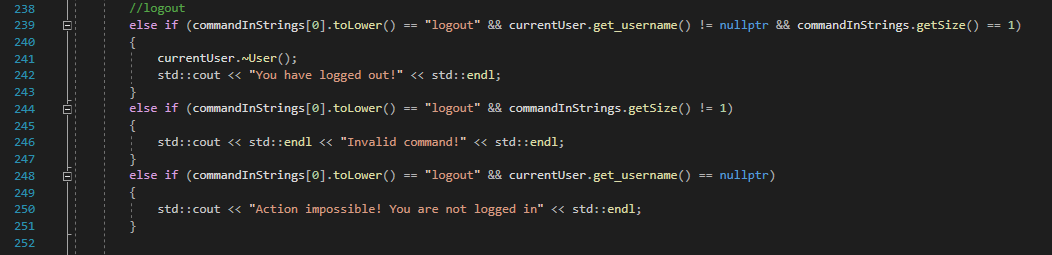
*Фиг. 3.4 Дефиниция на частни параметри и методи и публичен метод на класа Engine*

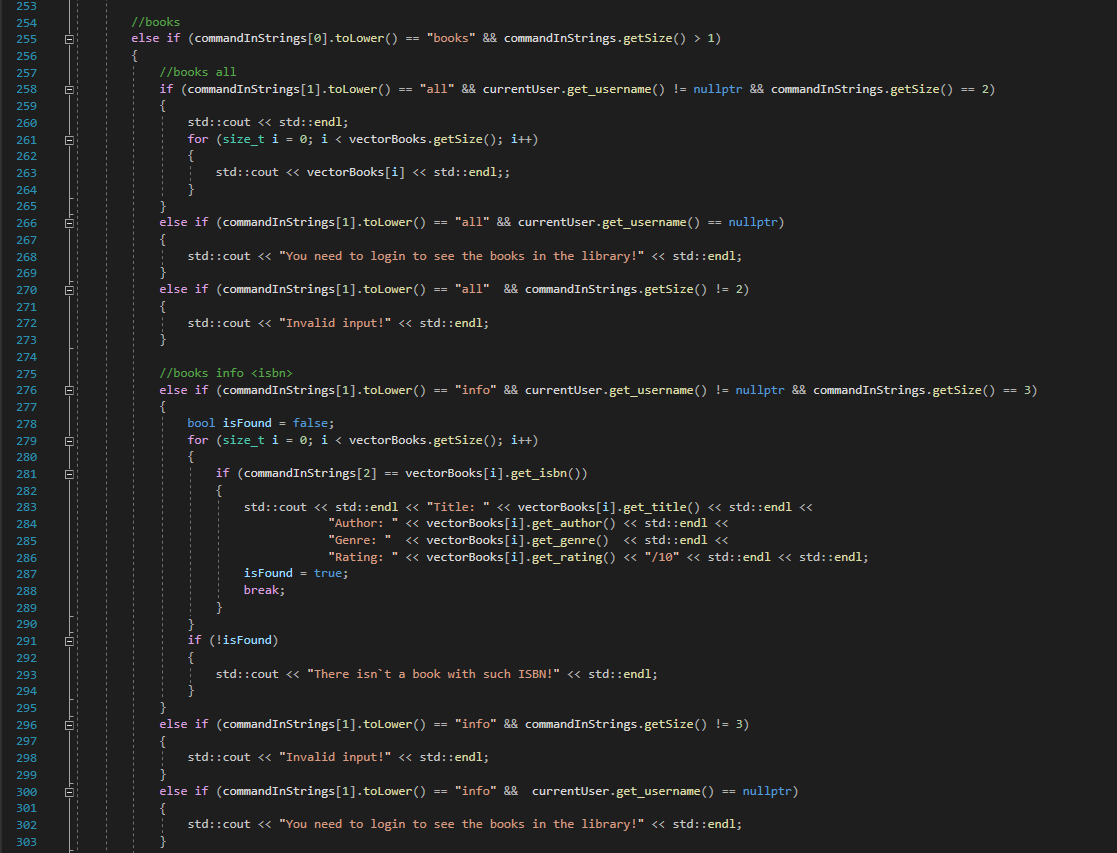
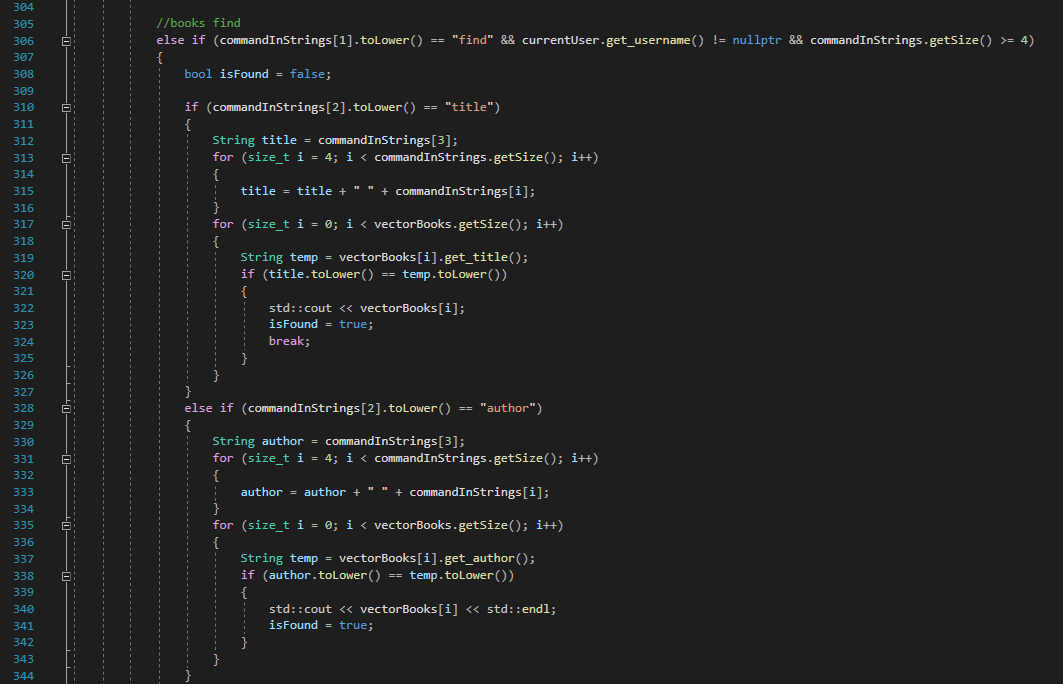
*Фиг. 3.5 Вход на команда, инициализация на цикъл за работа на меню. Създаване на потребител(1)*



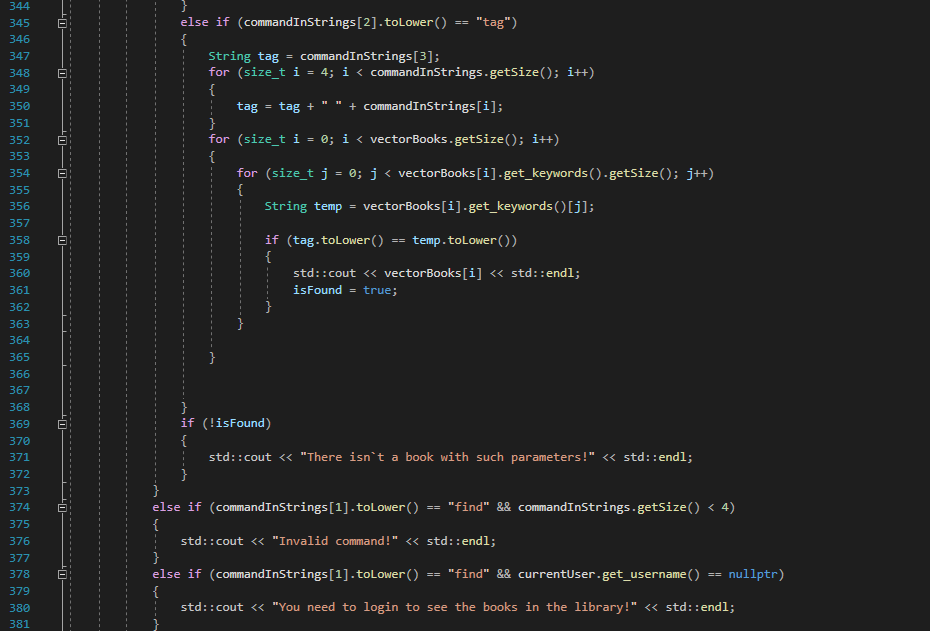
*Фиг. 3.6 Създаване на потребител (2). Изтриване на потребител*

*Фиг. 3.7 Всички потребители, влизане и излизане от потребителски профил*

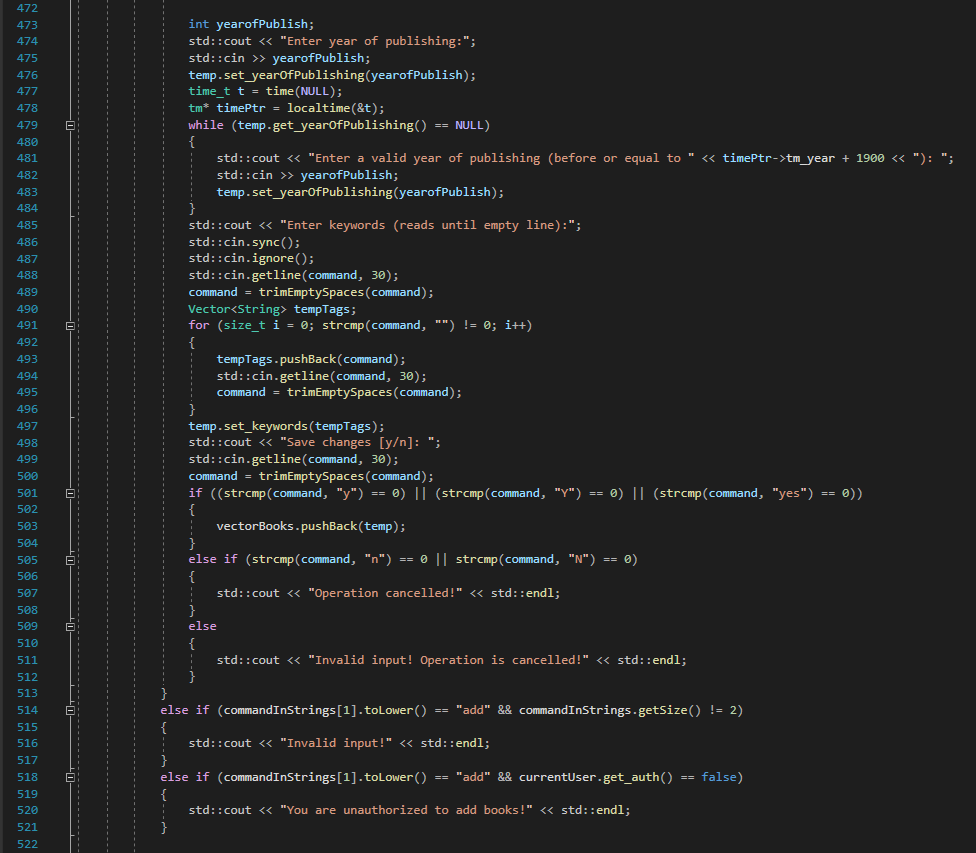
 

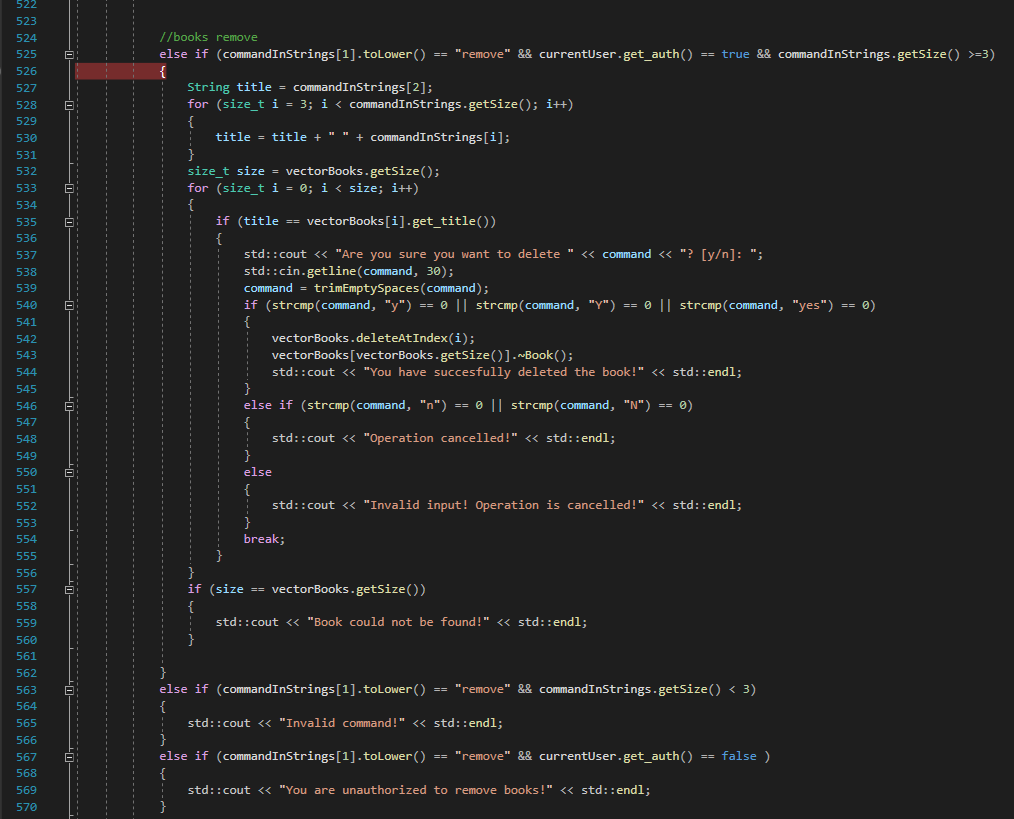
*Фиг. 3.8 Всички книги. Информация за една книга (по ISBN номер). Търсене на книга (1)*

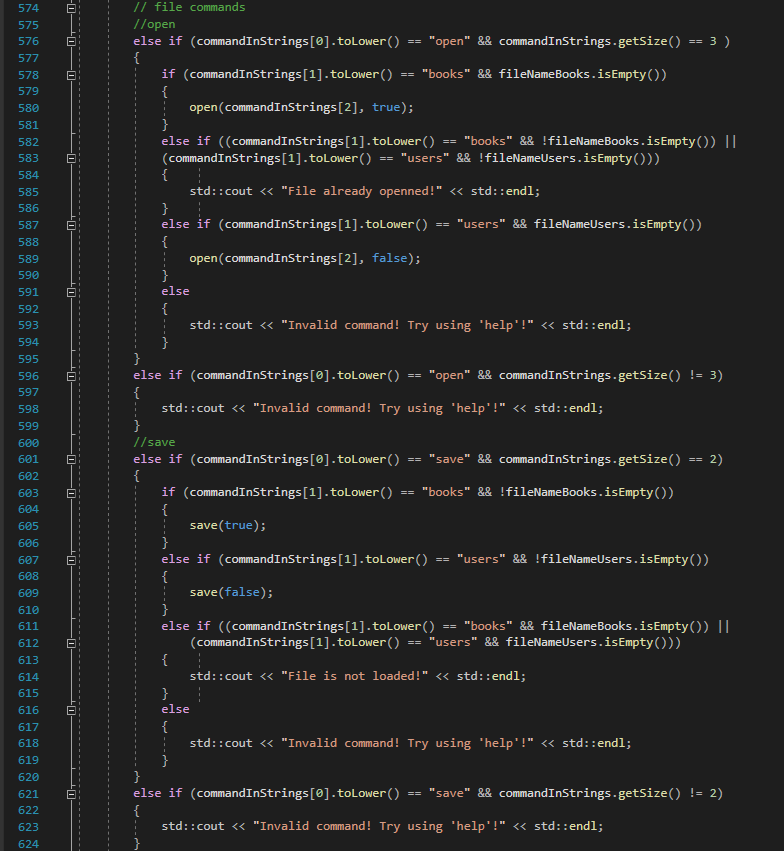
 

*Фиг. 3.9 Търсене на книга (2). Добавяне на книга (1)*

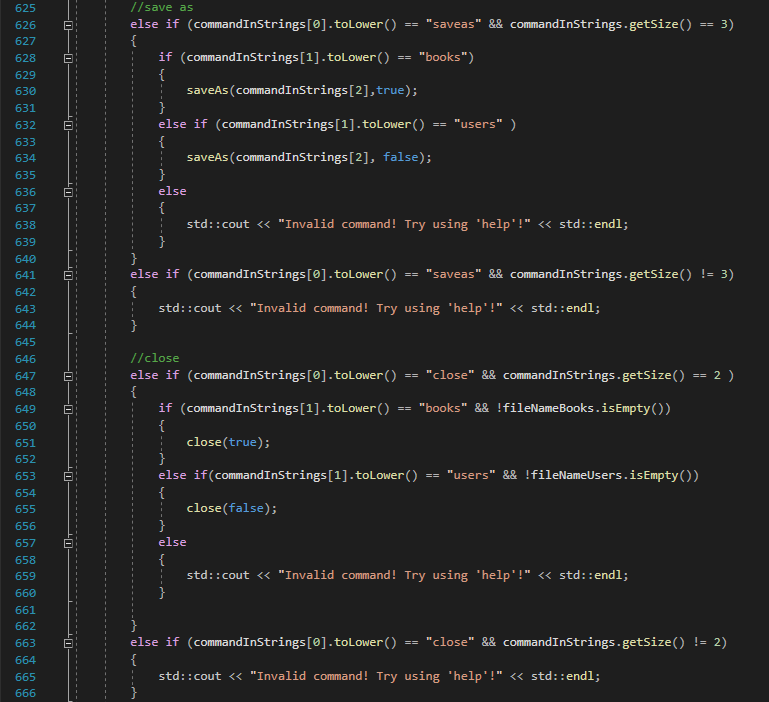
 *Фиг. 3.10 Добавяне на книга (2)*

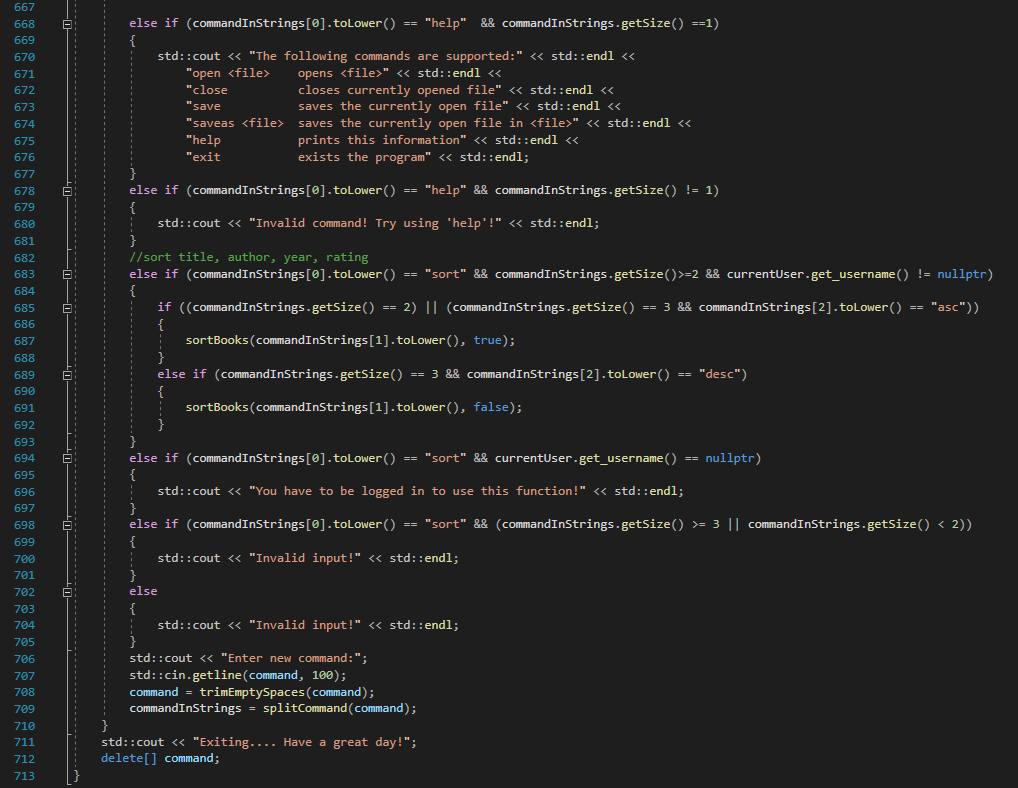
*Фиг. 3.11 Добавяне на книга (3)*

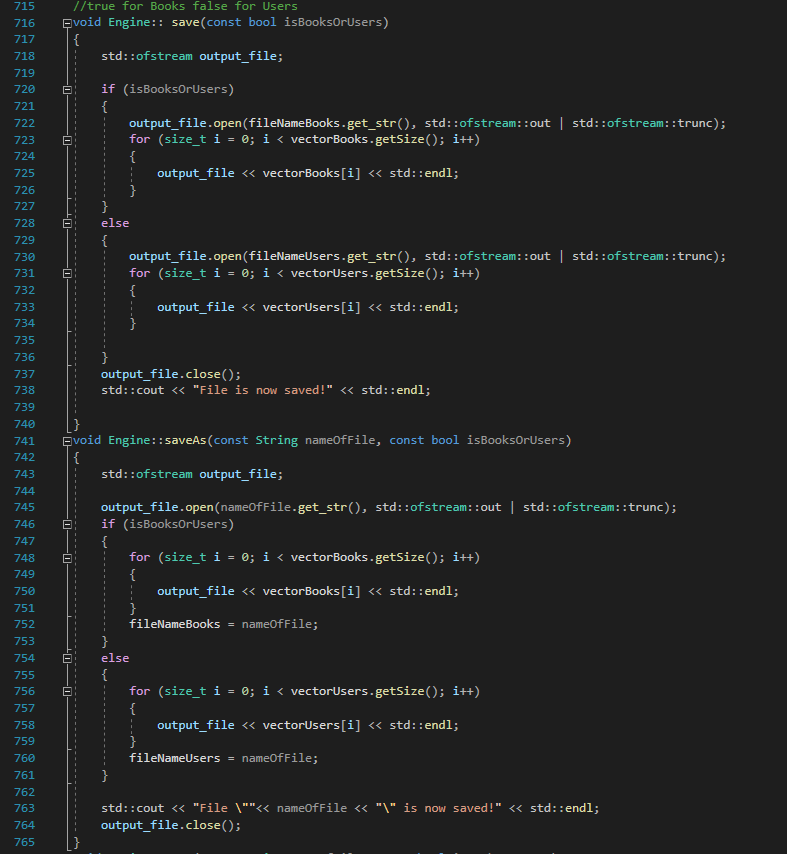
*Фиг. 3.12 Изтриване на книга*



*Фиг. 3.13 Работа с файлове (1)*

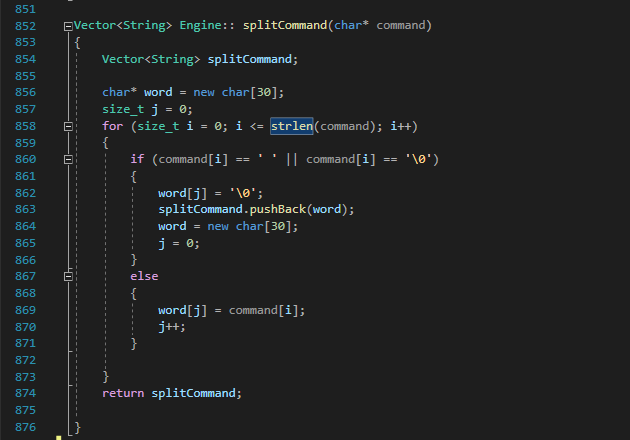
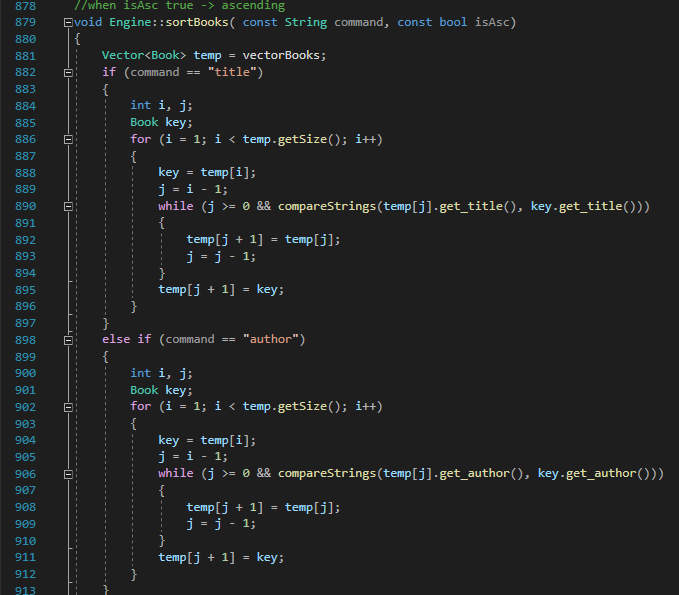
 *Фиг. 3.14 Работа с файлове (2)*

*Фиг. 3.15 „Помощ“, сортиране на книги, край на цикъла за меню*

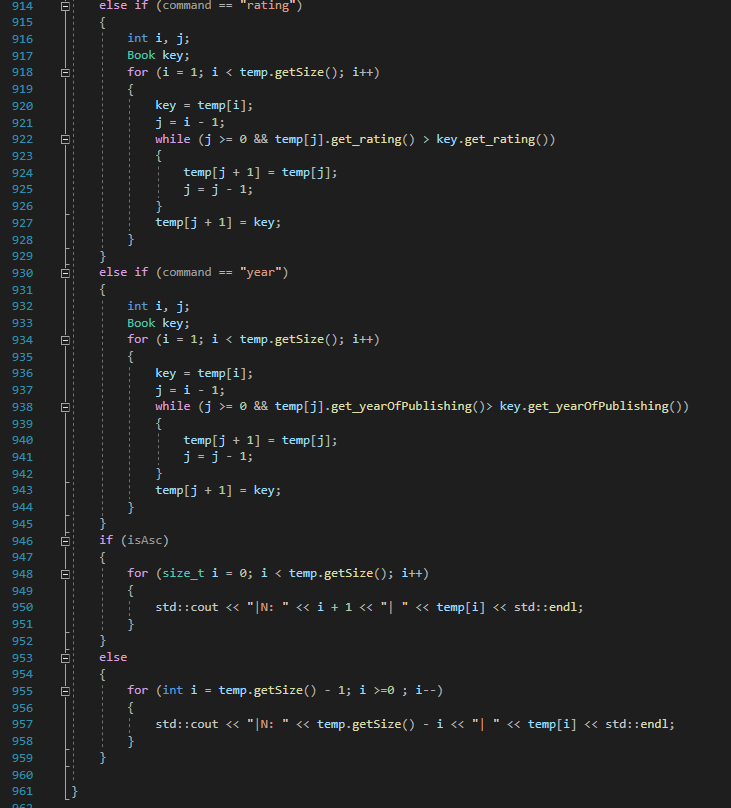
 *Фиг. 3.16 Запазване на отворен файл и запазване в указан файл*

*Фиг. 3.17 Отваряне на файл*

 *Фиг. 3.18 Затваряне на файл, изчистване на празни пространства преди и след команда*

*Фиг. 3.19 Разделяне на командата на думи*

*Фиг. 3.20 Сортиране на книги (1)*

 *Фиг. 3.21 Сортиране на книги(2)* 

*Фиг. 3.22 Сравняване на два масива от тип* ***char,*** *проверка за наличност на ISBN номер*

**Заключение**

В курсовата работа са реализирани основните изисквания, както са реализирани и допълнителни валидации и методи, които спомагат за качествената работа на програмата. Бъдещото развитие включва работа с реална база данни, вместо с текстови файлове, както и графичен интерфейс.

**Източници**

Презентации на Доцент, Доктор Калин Георгиев Иванов за курс по Обектно Ориентирано Програмиране на C++

**Съдържание**

Увод - стр. 2

Първа глава – Преглед на предметната област - стр. 3

Втора глава – Проектиране - стр. 7

Трета глава – Реализация и тестване - стр. 9

Заключение - стр. 29

Източници - стр. 29