

СУ "СВ. КЛИМЕТ ОХРИДСКИ"
ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА

КУРСОВ ПРОЕКТ №1

Тема: Електронна библиотека

Изготвил:
Ангел Владимиров Пенчев
спец. Компютърни науки
Курс: 1, Група: 2
Ф№: 5MI080026

гр. София, 2022

Съдържание

1	Въведение	3
1.1	Изисквания към реализацията на проекта	3
1.2	Технологии и развойни среди използвани за реализация на проекта	4
1.2.1	C++	4
1.2.2	CMake	4
1.2.3	gTest	5
1.2.4	Doxygen	5
1.2.5	Git	5
1.2.6	GitHub	5
1.2.7	GitHub Actions	6
1.2.8	CLion	6
2	Ръководство за потребителя	7
2.1	Изтегляне и изпълнение на компилирана версия на програмата	7
2.2	Настройка на среда за разработка, компилиране и изпълнение	7
2.3	Инструкции за употреба на проекта	9
3	Реализация на проекта	15
3.1	Структура на проекта	15
3.2	Програмна реализация на проекта	15
3.2.1	Клас Book	15

3.2.2	Клас User	21
3.2.3	Клас Library	24
3.2.4	Клас Program	33
3.2.5	Класове за грешки	34
3.2.6	Помощни класове	36
4	Заклучение	39
	Използвани източници	39

Глава 1.

Въведение

1.1 Изисквания към реализацията на проекта

Да се напише програма, реализираща информационна система, която поддържа библиотека от електронни книги. Програмата съхранява и обработва данни за наличните в момента книги в текстов или двоичен файл във формат по ваш избор. Всяка книга се характеризира със следните данни:

- автор
- заглавие
- име на текстов файл, където е записан текста на книгите
- кратко описание
- рейтинг
- международен стандартен номер на книга (ISBN)

Системата поддържа две нива на достъп: неоторизиран (за повечето операции) и оторизиран (за операции, които трябва да се

изпълняват с администраторски права). Достъпът до оторизираните функции става при въвеждане на парола за администратор.

Информацията за книгите в библиотеката се пази в текстов или двоичен файл във формат по ваш избор. Работата с програмата се осъществява в диалогов режим, като се поддържат следните операции:

Операция	
Добавяне на книга	
Премахване на книга	
Сортиране на списъка с книги	
Намиране на книга по критерий	Извежда на екрана подробна информация за
Извеждане на книга	Извежда на екрана

1.2 Технологии и развойни среди използвани за реализация на проекта

1.2.1 C++

C++ е език за програмиране от високо ниво, който има възможности за програмиране на големи програмни системи, с високо ниво на функционалност, представяне и ефикасност. Той е обектно ориентиран език със статични типове. Езикът се използва за изработката на проекта.

1.2.2 CMake

CMake е безплатен софтуер с отворен код за автоматизация на компилиране, тестване, пакетиране и инсталиране на софтуер чрез използване на независим от компилатора метод. CMake не е система за компилиране, а такава, която генерира компилационни файлове на други системи, като Make, Qt Creator, Ninja, Microsoft Visual

Studio и т.н. Инструментът се използва за пакетиране на програмата разработена в проекта.

1.2.3 gTest

Google Test е библиотека с отворен код за тестване на модули за езика C++. В проекта се използва за създаването и изпълняването на модулни тестове, интеграционни тестове и всеобхващащи (на англ. "end-to-end") тестове.

1.2.4 Doxygen

Doxygen е система за анализ и генериране на документация от коментари в изходния програмен код на даден проект. използвана е за генерацията на част от документацията, приложена към проекта.

1.2.5 Git

Git е децентрализирана система за контрол на версиите на файлове. Използва се за управление на версиите на проекта.

1.2.6 GitHub

GitHub е уеб базирана услуга разпространяване на софтуерни проекти, съвместни разработка върху тях и т.н. Проектите се съхраняват в т.нар. хранилища. Платформата се базира на Git системата за контрол и управление на версиите. Проектът се съхранява в GitHub кодово хранилище на адрес: <https://github.com/angel-penchev/librarity-but-dumber>.

1.2.7 GitHub Actions

GitHub Actions е платформа за непрекъсната интеграция и непрекъсната доставка (CI/CD), която позволява автоматизиране на пакетирането и тестването на проекти в платформата GitHub. В проекта се използва за изпълняването на тестове при качване в хранилището, както и за пакетирането на проекта при създаване на нова версия.

1.2.8 CLion

CLion е кросплатформенан развойна среда за разработка на C/C++ проекти. CLion включва такива функции като интелигентен редактор, генериране на код, осигуряване на качеството на кода, автоматизирано рефакториране, анализ на код, мениджмънт на проекти, интегрирани системи за контрол на версиите и дебъгер. CLion беше използвана за генерирането на структурата на проекта и разработката му.

Глава 2.

Ръководство за потребителя

2.1 Изтегляне и изпълнение на компилирана версия на програмата

Най-актуалната компилирана версия на проекта може да бъде достъпена в хранилището, на адрес: <https://github.com/angel-penchev/librarity-but-dumber/releases>. От там има възможност за изтеглянето на изпълним файл за Linux, Windows или macOS. След изтегляне, програмата може да се стартира, като за целта трябва да се изпълни на следната команда в директорията на изтегления файл:

1

```
./librarity_but_dumber
```

2.2 Настройка на среда за разработка, компилиране и изпълнение

За да се настрои среда за разработка на проекта, трябва да се изпълнят следните стъпки:

1. Да се изтегли кодовото хранилище на проекта от платформата GitHub.


```
1 git clone https://github.com/angel-penchev/librarity-but-dumber/  
2 cd librarity-but-dumber
```

2. Да се компилира проекта на проекта. Това може да бъде направено като се изпълни следната команда:

```
1 make
```

или ако инструмента Make е недостъпен:

```
1 # Create cmake configuration  
2 cmake -S. -Bcmake-build-debug  
   -DCMAKE_BUILD_TYPE=Debug~  
3  
4 # Build project  
5 cmake --build cmake-build-debug --config Debug
```

3. Да се изпълни проекта. Това може да бъде направено като се изпълни следната команда:

```
1 make run
```

или:

```
1 ./cmake-build-debug/librarity_but_dumber
```

За да се изпълнят тестовете на проекта, трябва да се изпълни следната команда:

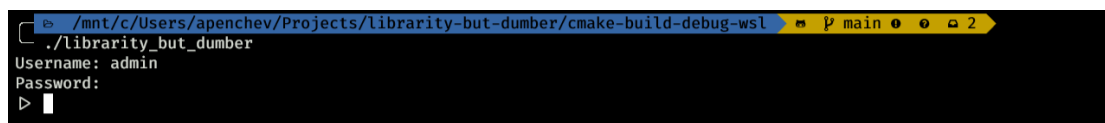
```
1 make tests
```

или:

```
1 cd cmake-build-debug  
2 ctest --test-dir -C Debug --output-on-failure --verbose
```

2.3 Инструкции за употреба на проекта

След стартиране на програмата, потребителят трябва да въведе име и парола за вход. Ако това е първото влизане в системата, потребителят трябва да въведе името "admin" и паролата "admin" за вход в администраторския профил. (Фиг. 2.1) Тези данни за автентикация могат да бъдат сменени по-късно.



Фигура 2.1: Вход в системата

След вход в системата, на потребителя се предоставя команден интерфейс, чрез който потребителя може да управлява информацията за книгите и потребителите. При получаване на командата "help" на потребителя се предоставя списък с всички команди, които могат да бъдат използвани. При получаване на командата "exit" или стандартна комбинация Ctrl+D за "приключване на входа програмата се затваря. Основните команди поддържани от системата са:

1. Команда за добавяне на потребител - "add user"

Изисква потребителят да има администраторски права. При извикване, изисква от потребителя да подаде на стандартния вход име и парола на потребителя, който ще бъде добавен в системата. Също така пита относно това, дали профила да има администраторски права или - не. (Фиг. 2.2)

```

C:\mnt\c\Users\apenchev\Projects\library-but-dumber\cmake-build-debug-wsl
./library_but_dumber
Username: admin
Password:
> add user
  Username: gosho
  Password:
  Is administrator (y/n): n
>

```

Фигура 2.2: Добавяне на потребител в системата

2. Команда за смяна на паролата - "change password"

При извикване, изисква от потребителя старата му парола, нова парола и повторена нова парола. При несъответствие на паролите, потребителят се показва съобщение за грешка. При валидно промяна на паролата, потребителят се показва съобщение за успешно промяна на паролата. (Фиг. 2.3)

```

C:\mnt\c\Users\apenchev\Projects\library-but-dumber\cmake-build-debug-wsl
./library_but_dumber
Username: admin
Password:
> change password
  Old password:
  New password:
  New password (confirm):
>

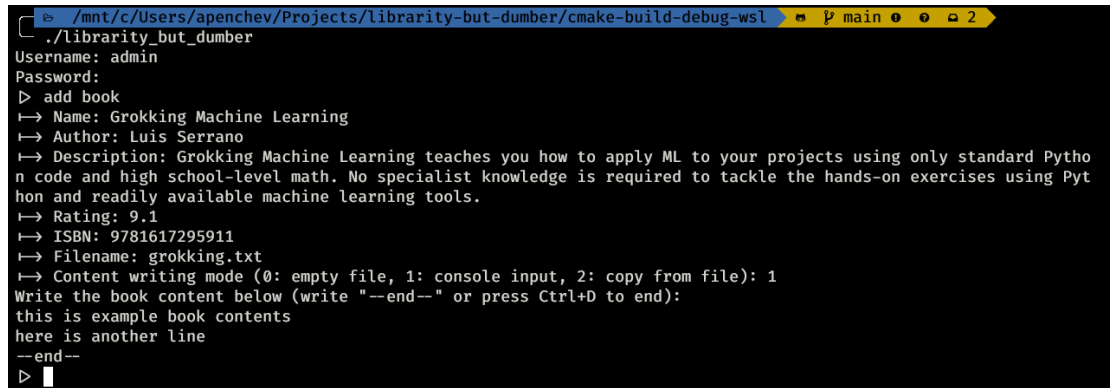
```

Фигура 2.3: Смяна на потребителската парола

3. Команда за добавяне на книга - "add book"или "add"


Изисква потребителят да има администраторски права. При извикване, потребителят трябва да подаде на стандартния вход всичката информация нужна за създаването на книга - име, автор, описание, рейтинг, международен стандартен номер (ISBN) и файлов път, където ще се запази текста на книгата. Ако на файловия път вече съществува файл, потребителя се запитва дали да презапише съдържанието му. След това, на потребителя се дава възможност да избере начин за

създаване на файла със съдържанието на книгата - да създаде празен файл, да създаде файл и да го попълни от стандартния вход (Фиг. 2.4) или да създаде файл и да го попълни със съдържанието на друг файл. (Фиг. 2.5)



```
./library_but_dumber
Username: admin
Password:
> add book
  ↳ Name: Grokking Machine Learning
  ↳ Author: Luis Serrano
  ↳ Description: Grokking Machine Learning teaches you how to apply ML to your projects using only standard Python code and high school-level math. No specialist knowledge is required to tackle the hands-on exercises using Python and readily available machine learning tools.
  ↳ Rating: 9.1
  ↳ ISBN: 9781617295911
  ↳ Filename: grokking.txt
  ↳ Content writing mode (0: empty file, 1: console input, 2: copy from file): 1
Write the book content below (write "--end--" or press Ctrl+D to end):
this is example book contents
here is another line
--end--
>
```

Фигура 2.4: Добавяне на книга в системата със съдържание от стандартния вход



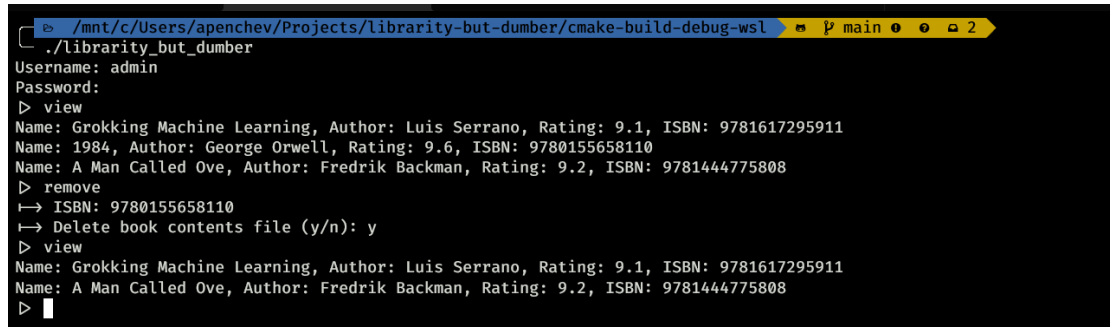
```
./library_but_dumber
Username: admin
Password:
WARN: Invalid password! You have 2 attempts left.
Password:
> add
  ↳ Name: 1984
  ↳ Author: George Orwell
  ↳ Description: Written more than 70 years ago, 1984 was George Orwell's chilling prophecy about the future. And while 1984 has come and gone, his dystopian vision of a government that will do anything to control the narrative is timelier than ever...
  ↳ Rating: 9.6
  ↳ ISBN: 9780155658110
  ↳ Filename: 1984.txt
  ↳ Content writing mode (0: empty file, 1: console input, 2: copy from file): 2
  ↳ Source filename: grokking.txt
>
```

Фигура 2.5: Добавяне на книга в системата със съдържание от текстов файл

4. Команда за премахване на книга - "remove book"или "remove"

Изисква потребителят да има администраторски права. Пита потребителя да въведе идентификационния номер (ISBN) на

книгата, която да бъде премахната и дали иска да премахне файла със съдържанието на книгата. (Фиг. 2.6)



```

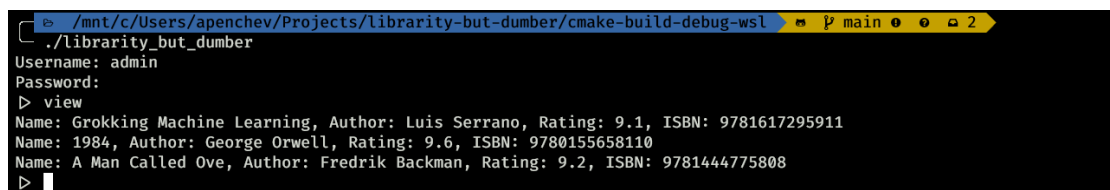
/mnt/c/Users/apenchev/Projects/librarity-but-dumber/cmake-build-debug-wsl
./librarity_but_dumber
Username: admin
Password:
> view
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911
Name: 1984, Author: George Orwell, Rating: 9.6, ISBN: 9780155658110
Name: A Man Called Ove, Author: Fredrik Backman, Rating: 9.2, ISBN: 9781444775808
> remove
  ↳ ISBN: 9780155658110
  ↳ Delete book contents file (y/n): y
> view
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911
Name: A Man Called Ove, Author: Fredrik Backman, Rating: 9.2, ISBN: 9781444775808
> █

```

Фигура 2.6: Премахване на книга от системата

5. Команда за преглед на списък с книгите в системата - "list books" или "list" или "view"

Показва всички настоящо добавени книги в системата в не-подреден вид. Не изисква допълнителни параметри. (Фиг. 2.7)



```

/mnt/c/Users/apenchev/Projects/librarity-but-dumber/cmake-build-debug-wsl
./librarity_but_dumber
Username: admin
Password:
> view
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911
Name: 1984, Author: George Orwell, Rating: 9.6, ISBN: 9780155658110
Name: A Man Called Ove, Author: Fredrik Backman, Rating: 9.2, ISBN: 9781444775808
> █

```

Фигура 2.7: Преглед на списък с книгите в системата

6. Команда за сортиране на книгите в системата - "sort books" или "sort"

Сортира и показва всички настоящо добавени книги в системата. Изисква от потребителя да избере критерий за сортиране - по име, по автор или по рейтинг. (Фиг. 2.8)

```

/mnt/c/Users/apenchev/Projects/librarity-but-dumber/cmake-build-debug-wsl
./librarity_but_dumber
Username: admin
Password:
> sort
↳ Sorting mode (0: by name, 1: by author, 2: by rating): 0
Name: 1984, Author: George Orwell, Rating: 9.6, ISBN: 9780155658110
Name: A Man Called Ove, Author: Fredrik Backman, Rating: 9.2, ISBN: 9781444775808
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911
>

```

Фигура 2.8: Сортиране на книгите в системата

7. Команда за намиране на книга в системата - "find book"или "find"

Пита потребителя да избере по кое поле на книгата да търси - име, автор, ISBN или част от описание. След това изисква да се въведе символен низ, по който да се търси в съответното поле. Важно е да се отбележи, че капитализацията на низа не е от значение. (Фиг. 2.9)

```

/mnt/c/Users/apenchev/Projects/librarity-but-dumber/cmake-build-debug-wsl
./librarity_but_dumber
Username: admin
Password:
> find
↳ Find mode (0: by name, 1: by author, 2: by ISBN, 3: by description snippet): 3
↳ Description snippet: machine
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911
>

```

Фигура 2.9: Намеране на книга в системата

8. Команда за принтиране на книга в системата - "print"

Пита потребителя да въведе идентификационния номер (ISBN) на книгата, която да бъде принтирана. След това изисква да се избере 1 от 3 режима за принтиране - на цялото съдържание, на брой редове, зададени от потребителя, след които се пазира и на отделни изречения. (Фиг. 2.10)

```

./library_but_dumber
Username: admin
Password:
> print
↳ ISBN: 9781617295911
↳ Reading mode (0: whole book, 1: pages, 2: sentences): 1
↳ Lines per page: 6
Name: Grokking Machine Learning, Author: Luis Serrano, Rating: 9.1, ISBN: 9781617295911

We're no strangers to love
You know the rules and so do I (do I)
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand

-- press enter to continue --

Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
Never gonna tell a lie and hurt you

-- press enter to continue --

```

Фигура 2.10: Принтиране на книга в системата

Глава 3.

Реализация на проекта

3.1 Структура на проекта

3.2 Програмна реализация на проекта

3.2.1 Клас Book

Класът "Book" съдържа в себе си полета за име, автор, описание, име, автор, описание, рейтинг, международен стандартен номер (ISBN) и файлов път, където ще се запази текста на книгата.

```
1 class Book {  
2 private:  
3     char *name;  
4     char *author;  
5     char *description;  
6     double rating;  
7     char *ISBN;  
8     char *filename;  
9     ...  
10 }
```

Класът има стандартен конструктор, копиращ конструктор, оператор за равенство и деструктор, които изпълняват каноничната форма, нужна при заделяне на динамична памет в класа.


```

1 class Book {
2     ...
3 public:
4     explicit Book(const char *name = "Untitled", const char *author =
5         "Unknown_author", const char *description = "",
6         double rating = 0, const char *ISBN = "9780000000002", const char
7         *filename = "");
8
9     Book(const Book &other);
10
11    Book &operator=(const Book &other);
12
13    virtual ~Book();
14    ...
15 };

```

"Book също така, предоставя селектори и мутатори за промяна на полетата на класа, които заделят динамична памет, ако това е нужно.

```

1 class Book {
2     ...
3 public:
4     ...
5     char *getName() const;
6
7     void setName(const char *newName);
8
9     char *getAuthor() const;
10
11    void setAuthor(const char *newAuthor);
12
13    char *getDescription() const;
14
15    void setDescription(const char *newDescription);
16
17    double getRating() const;

```

```

18
19     void setRating(double newRating);
20
21     char *getISBN() const;
22
23     void setISBN(const char *newISBN);
24
25     char *getFilename() const;
26
27     void setFilename(const char *newFilename);
28     ...
29 };

```

Класът предефинира оператора « при принтиране и писане по изходен двоичен файлов поток.

```

1 class Book {
2     ...
3 public:
4     ...
5     friend std::ostream &operator<<(std::ostream &os, const Book &book);
6
7     friend std::ofstream &operator<<(std::ofstream &out, const Book &book);
8     ...
9 };

```

Класът реализира три метода за принтиране на съдържанието на книгите - метод принтиращ цялата книга наведнъж, метод принтиращ задаен брой редове и изчаквайки потвърждение да продължи и метод принтиращ отделни изречения от книгата.

```

1 /**
2  * Prints the whole book contents file .
3  */
4 void Book::printAllContents() const {
5     // Try to open contents file for reading
6     std::ifstream booksContentsFile(this->filename, std::ios::in);
7     if (!booksContentsFile) {
8         throw

```

```

        BookException(BookErrorCode::CONTENTS_FILE_READING_ERR);
9     }
10
11     // Read every line and print it until the end of the file
12     while (!booksContentsFile.eof()) {
13         char line[MAX_LINE_LEN];
14         booksContentsFile.getline(line, MAX_LINE_LEN);
15         std::cout << line << '\n';
16     }
17
18     // Close the contents file
19     booksContentsFile.close();
20 }
21
22 /**
23  * Prints a certain amount of lines from the book contents file and waits for
24  * input confirmation to continue.
25  * @param linesCount Amount of lines to print before pause
26  */
27 void Book::printPaginatedContents(unsigned int linesCount) const {
28     // Try to open contents file for reading
29     std::ifstream booksContentsFile(this->filename, std::ios::in);
30     if (!booksContentsFile) {
31         throw
32             BookException(BookErrorCode::CONTENTS_FILE_READING_ERR);
33     }
34
35     // Read and print N lines at a time until the end of the file
36     while (!booksContentsFile.eof()) {
37         for (unsigned int i = 0; i < linesCount; i++) {
38             char line[MAX_LINE_LEN];
39             booksContentsFile.getline(line, MAX_LINE_LEN);
40             std::cout << line << '\n';
41         }
42
43         std::cout << "\n--_press_enter_to_continue_--\n";
44         std::cin.ignore();
45     }

```

```

44
45     // Close the contents file
46     booksContentsFile.close();
47 }
48
49 /**
50  * Prints the book 1 sentence at a time, waiting for input confirmation to
51  * continue.
52  */
53 void Book::printSentenceSeparatedContents() const {
54     // Try to open contents file for reading
55     std::ifstream booksContentsFile(this->filename, std::ios::in);
56     if (!booksContentsFile) {
57         throw
58             BookException(BookErrorCode::CONTENTS_FILE_READING_ERR);
59     }
60
61     // Read and print the contents, pausing at sentence separators
62     while (!booksContentsFile.eof()) {
63         char line[MAX_LINE_LEN];
64         booksContentsFile.getline(line, MAX_LINE_LEN, '.');
65         std::cout << line << "\n\n--_press_enter_to_continue_--\n";
66         std::cin.ignore();
67     }
68
69     // Close the contents file
70     booksContentsFile.close();
71 }

```

Класът реализира методи за обновяване на текста на книгата. Поддържат се два метода на въвеждане - от стандартния вход и от текстови файл.

```

1  /**
2  * Updates the contents file of the Book from a string.
3  * @param line Line to be written
4  * @param isTruncateMode Whether to delete previous file contents or not
5  */

```

```

6 void Book::updateContents(const char *line, bool isTruncateMode) const {
7     // Open contents file for writing
8     std::ofstream booksContentsFile(
9         this->filename, std::ios::out | (isTruncateMode ? std::ios::trunc
10             : std::ios::app));
11     if (!booksContentsFile) {
12         throw
13             BookException(BookErrorCode::CONTENTS_FILE_WRITING_ERR);
14     }
15
16     // Write the text input to the text file
17     unsigned int lineLength = std::strlen(line) + 1;
18     booksContentsFile.write(line, lineLength) << '\n';
19
20     // Close the contents file
21     booksContentsFile.close();
22 }
23
24 /**
25  * Updates the contents file of the Book from a text file input stream.
26  * @param input Input stream to copy from
27  * @param isTruncateMode Whether to delete previous file contents or not
28  */
29 void Book::updateContents(std::ifstream &input, bool isTruncateMode) const {
30     // Open contents file for writing
31     std::ofstream booksContentsFile(
32         this->filename, std::ios::out | (isTruncateMode ? std::ios::trunc
33             : std::ios::app));
34     if (!booksContentsFile) {
35         throw BookException(CONTENTS_FILE_WRITING_ERR);
36     }
37
38     // Copy the contents of the input file to the book contents file
39     booksContentsFile << input.rdbuf();
40
41     // Close the contents file
42     booksContentsFile.close();
43 }

```

Класът съдържащ информация за книгата, предоставя и методи за триене на съдържанието и от файловата система.

```
1  /**
2   * Deletes the book contents file
3   */
4  void Book::deleteBookContents() const {
5      // Remove contents file
6      if (remove(this->filename)) {
7          throw
8              BookException(BookErrorCode::CONTENTS_FILE_REMOVAL_ERR);
9      }
10 }
```

"Book" предоставя публични методи за валидация на данните на книгата. Поддържат се два метода за валидация - на ISBN и на рейтинг.

```
1  class Book {
2  private:
3      ...
4  public:
5      ...
6      static void validateRating(double newRating);
7
8      static void validateISBN(const char *newISBN);
9
10     static bool isValidRating(double newRating);
11
12     static bool isValidISBN(const char *newISBN);
13 };
```

3.2.2 Клас User

Класът "User" съдържа в себе си полета за потребителско име, потребителска парола и за това дали има администраторски пра-

ВОМОЩИЯ.

```
1 class User {  
2 private:  
3     char *username;  
4     char *passwordHash;  
5     bool isAdmin;  
6     ...  
7 }
```

Класът има стандартен конструктор, копиращ конструктор, оператор за равенство и деструктор, които изпълняват каноничната форма, нужна при заделяне на динамична памет в класа.

```
1 class User {  
2 private:  
3     ...  
4 public:  
5     explicit User(const char *username = "", const char *password = "", bool  
6         isAdmin = false);  
7     User(const User &other);  
8  
9     virtual ~User();  
10    ...  
11 };
```

"User също така, предоставя селектори и мутатори за промяна на полетата на класа, които заделят динамична памет, ако това е нужно.

```
1 class User {  
2 private:  
3     ...  
4  
5 public:  
6     ...  
7  
8     char *getUsername() const;  
9
```

```

10     void setUsername(const char *newUsername);
11
12     void setPassword(const char *newPassword, bool isEncrypted = false);
13
14     bool isAdministrator() const;
15
16     void setIsAdministrator(bool newIsAdmin);
17 };

```

Класът реализира метод за сериализация на съдържанието му във файл.

```

1  /**
2   * Output "<<" operator override for output file streams.
3   * Serializes the information of a User in a binary output file .
4   * @param out Output file stream
5   * @param book Reference to a Book object to output
6   * @return
7   */
8  std::ofstream &operator<<(std::ofstream &out, const User& user) {
9      unsigned int usernameLength = std::strlen(user.username) + 1;
10     out.write((const char *) &usernameLength, sizeof(usernameLength));
11     out.write((const char *) user.username, usernameLength);
12
13     unsigned int passwordHashLength = std::strlen(user.passwordHash) + 1;
14     out.write((const char *) &passwordHashLength,
15              sizeof(passwordHashLength));
16     out.write((const char *) user.passwordHash, passwordHashLength);
17
18     out.write((const char *) &user.isAdmin, sizeof(user.isAdmin));
19
20     return out;
21 }

```

Също така има метод за валидация на въведена потребителска парола.

```

1  /**
2   * Verifies whether an input password matches the one User has.

```



```

3  * @param password Password to be checked
4  * @return Whether the passwords match or not
5  */
6  bool User::verifyPassword(const char *password) {
7      const char *encryptedPassword = encryptPassword(password);
8      return !std::strcmp(encryptedPassword, this->passwordHash);
9  }

```

3.2.3 Клас Library

Класът "User" съдържа в себе си два списъка - 1 за потребители и 1 за книги, както и полета за техните размери и имената на файловете, в които се запазват.

```

1  class Library {
2  private:
3      Book *books;
4      User *users;
5      char *booksFilename;
6      char *usersFilename;
7      unsigned int booksCount = 0;
8      unsigned int usersCount = 0;
9      ...
10 }

```

Класът има стандартен конструктор, копиращ конструктор, оператор за равенство и деструктор, които изпълняват каноничната форма, нужна при заделяне на динамична памет в класа.

```

1  class Library {
2  private:
3      ...
4  public:
5      Library();
6
7      Library(const Library &other);
8
9      Library &operator=(const Library &other);

```

```

10
11     virtual ~Library();
12 };

```

За да се конструира "Library" в Program се използва конструктор приемащ като аргументи пътищата на файловете за записване на книги и на потребители.

```

1  /**
2   * Library parameter constructor.
3   * @param booksFilename Filename of the books binary file
4   * @param usersFilename Filename of the users binary file
5   */
6  Library::Library(const char *booksFilename, const char *usersFilename) :
7      books(new Book[0]), users(new User[0]),
8      booksFilename(),
9      usersFilename()
10     {
11
12     // Set books and users filename properties
13     this->setBooksFilename(booksFilename);
14     this->setUsersFilename(usersFilename);
15
16     // Read books from the books binary if it exists
17     std::ifstream booksFile(this->booksFilename, std::ios::binary |
18         std::ios::in);
19     if (booksFile) {
20         unsigned int booksCountFromFile;
21         booksFile.read((char *) &booksCountFromFile,
22             sizeof(booksCountFromFile));
23         for (unsigned int i = 0; i < booksCountFromFile; i++) {
24             this->addBook(Book(booksFile));
25         }
26     }
27     booksFile.close();
28
29     // Try to open users file and read the users from there.
30     // If that fails, creating a new users file with a default admin.
31     std::ifstream usersFile(usersFilename, std::ios::binary | std::ios::in);

```

```

26     if (!usersFile) {
27         // Add a default administrator to the users
28         User defaultAdmin = User("admin", "admin", true);
29         this->addUser(defaultAdmin);
30
31         // Create new users file with the administrator
32         this->updateUsersFile();
33         return;
34     }
35
36     // Read users from the books binary
37     unsigned int usersCountFromFile;
38     usersFile.read((char *) &usersCountFromFile, sizeof(usersCountFromFile));
39     for (unsigned int i = 0; i < usersCountFromFile; i++) {
40         this->addUser(User(usersFile));
41     }
42     usersFile.close();
43 }

```

"Library" поддържа методи за добавяне на потребители и книги:

```

1  /**
2   * Adds a book to the books array.
3   * @param book Book to be added
4   * @return Reference to the added book
5   */
6  Book *Library::addBook(const Book &book) {
7      // Verify a book with the same ISBN doesn't exist
8      if (this->findBookIndex(book.getISBN(), FindMode::FIND_BY_ISBN)
9          >= 0) {
10         throw LibraryException(LibraryErrorCode::DUPLICATE_ISBN);
11     }
12
13     // Allocate a new array with increased size
14     Book *newArr = new Book[this->booksCount + 1];
15
16     // Copy all book to the new array and add the new one to the end
17     for (unsigned int i = 0; i < this->booksCount; i++) {
18         newArr[i] = this->books[i];
19     }
20     newArr[this->booksCount] = book;
21     delete [] books;
22     books = newArr;
23 }

```

```

18     }
19     newArr[this->booksCount] = book;
20
21     // Delete the old array and set the object one
22     delete [] this->books;
23     this->books = newArr;
24
25     // Return a reference to the added book
26     return &this->books[this->booksCount++];
27 }
28
29 /**
30  * Adds an user to the
31  * @param user User to be added
32  * @return Reference to the added user
33  */
34 User *Library::addUser(const User &user) {
35     // Verify a user with the same username doesn't exist
36     if (this->findUserIndex(user.getUsername()) >= 0) {
37         throw
38             LibraryException(LibraryErrorCode::DUPLICATE_USERNAME);
39     }
40
41     // Allocate a new array with increased size
42     User *newArr = new User[this->usersCount + 1];
43
44     // Copy all users to the new array and add the new one to the end
45     for (unsigned int i = 0; i < this->usersCount; i++) {
46         newArr[i] = this->users[i];
47     }
48     newArr[this->usersCount] = user;
49
50     // Delete the old array and set the object one
51     delete [] this->users;
52     this->users = newArr;
53
54     // Return a reference to the added user
55     return &this->users[this->usersCount++];

```

55

}

и за обновяване на съответните им файлове:

```

1  /**
2   * Updates the books binary file with the current books array content.
3   */
4  void Library::updateBooksFile() const {
5      std::ofstream booksFile(this->booksFilename, std::ios::binary |
6          std::ios::out | std::ios::trunc);
7      booksFile.write((const char *) &this->booksCount,
8          sizeof(this->booksCount));
9      for (unsigned int i = 0; i < this->booksCount; i++) {
10         booksFile << this->books[i];
11     }
12     booksFile.close();
13 }
14
15 /**
16 * Updates the user binary file with the current books array content.
17 */
18 void Library::updateUsersFile() const {
19     std::ofstream usersFile(this->usersFilename, std::ios::binary |
20         std::ios::out | std::ios::trunc);
21     usersFile.write((const char *) &this->usersCount,
22         sizeof(this->usersCount));
23     for (unsigned int i = 0; i < this->usersCount; i++) {
24         usersFile << this->users[i];
25     }
26     usersFile.close();
27 }

```

Класът също така има метод за изтриване на книги:

```

1  /**
2   * Delete a book from books.
3   * @param ISBN ISBN identifier of the book to be deleted
4   * @param deleteContentsFile Whether to delete the contents file or not
5   */
6  void Library::removeBook(const char *ISBN, const bool deleteContentsFile) {

```

```

7 // Try to find the book index in books
8 int bookIndex = this->findBookIndex(ISBN,
   FindMode::FIND_BY_ISBN);
9
10 // If the index is less than 0 -> the book was not found
11 if (bookIndex < 0) {
12     throw
        LibraryException(LibraryErrorCode::BOOK_NOT_FOUNT_ERR);
13 }
14
15 // Delete the contents file if requested
16 if (deleteContentsFile) {
17     this->books[bookIndex].deleteBookContents();
18 }
19
20 // Create a new array with one space removed
21 Book *newArr = new Book[--this->booksCount];
22
23 // Copy all books before the index in the new array
24 for (unsigned int i = 0; i < bookIndex; i++) {
25     newArr[i] = this->books[i];
26 }
27
28 // Copy all books after the index in the new array
29 for (unsigned int i = bookIndex; i < this->booksCount; i++) {
30     newArr[i] = this->books[i + 1];
31 }
32
33 // Delete the old array and set the object one
34 delete [] this->books;
35 this->books = newArr;
36 }

```

за сортирането им:

```

1 /**
2  * Sort books by criteria .
3  * @param sortingMode
4  */

```

```

5 void Library::sortBooks(SortingMode sortingMode) {
6     for (int i = 0; i < (int) this->booksCount - 1; i++) {
7         for (int j = i + 1; j < (int) this->booksCount; j++) {
8             bool swapCondition;
9             switch (sortingMode) {
10                case SortingMode::SORT_BY_NAME:
11                    swapCondition = std::strcmp(this->books[i].getName(),
12                                                this->books[j].getName()) > 0;
13                    break;
14                case SortingMode::SORT_BY_AUTHOR:
15                    swapCondition = std::strcmp(this->books[i].getAuthor(),
16                                                this->books[j].getAuthor()) > 0;
17                    break;
18                case SortingMode::SORT_BY_RATING:
19                    swapCondition = this->books[i].getRating() >
20                                this->books[j].getRating();
21                    break;
22            }
23            if (swapCondition) {
24                Book tempBook = books[i];
25                this->books[i] = this->books[j];
26                this->books[j] = tempBook;
27            }
28        }
29    }
30 }

```

за търсенето им:

```

1 /**
2  * Case-insensitive books search by criteria .
3  * @param query
4  * @param findMode Whether to search by name, author, ISBN or description
5  * snippet
6  * @return Pointer to the book found
7  */
8 Book *Library::findBook(const char *query, FindMode findMode) const {
9     // Try to find the book index in books
10 }

```

```

9      int bookIndex = this->findBookIndex(query, findMode, false);
10
11      // If the index is less than 0 -> the book was not found
12      if (bookIndex < 0) {
13          return nullptr;
14      }
15
16      // Return a pointer to the book
17      return &this->books[bookIndex];
18  }

```

и за принтирането им:

```

1  /**
2   * Prints the contents of a book.
3   * @param book Book whose contents to be printed
4   * @param readingMode Whether to print the entire book at once, with page
5   *   separation or with sentence separation.
6   * @param linesCount Lines in a book page
7   */
8  void Library::printBookContent(Book *book, ReadingMode readingMode,
9      unsigned int linesCount) {
10      switch (readingMode) {
11          case ReadingMode::WHOLE_BOOK:
12              book->printAllContents();
13              break;
14          case ReadingMode::PAGES:
15              book->printPaginatedContents(linesCount);
16              break;
17          case ReadingMode::SENTENCES:
18              book->printSentenceSeparatedContents();
19              break;
20          default:
21              throw std::invalid_argument("Unimplemented_item");
22      }
23  }

```

Класът също така съдържа функции за вход на потребители и пормяна на паролите им:


```

1  /**
2   * Attempt to login a user with username and password credentials.
3   * @param username Username to attempt login with
4   * @param password Password to attempt login with
5   * @return Pointer to the user on successful login , otherwise nullptr
6   */
7  User *Library::loginUser(const char *username, const char *password) const {
8      // Find the user by username
9      int userIndex = this->findUserIndex(username);
10
11      // If the user is not found or if the password is incorrect , return
12      // nullptr
13      if (userIndex < 0 || !this->users[userIndex].verifyPassword(password)) {
14          return nullptr;
15      }
16
17      // Return the user if login credentials are correct
18      return &this->users[userIndex];
19  }
20
21  /**
22   * Change user password.
23   * @param username User username
24   * @param oldPassword Old user password
25   * @param newPassword New password to be set
26   * @param newPasswordConfirm Confirmation of the new password
27   */
28  void Library::changeUserPassword(const char *username, const char
29      *oldPassword, const char *newPassword,
30      const char *newPasswordConfirm) const {
31      // Find the user by username
32      int userIndex = this->findUserIndex(username);
33      if (userIndex < 0) {
34          throw
35              LibraryException(LibraryErrorCode::BOOK_NOT_FOUNT_ERR);
36      }
37  }

```

```

35
36 // Verifying the old password matches the one stored
37 if (!this->users[userIndex].verifyPassword(oldPassword)) {
38     throw
39         LibraryException(LibraryErrorCode::INVALID_OLD_PASSWORD);
40 }
41
42 // Verifying the new password and its confirmation match
43 if (std::strcmp(newPassword, newPasswordConfirm) != 0) {
44     throw
45         LibraryException(LibraryErrorCode::MISMATCHING_PASSWORDS_ERR);
46 }
47
48 // Change the password
49 this->users[userIndex].setPassword(newPassword);
50
51 // Update users file after the change
52 this->updateUsersFile();
53 }

```

3.2.4 Клас Program

Класът "Program" съдържа в себе си публичният статичен метод `run()`, който предоставя формата за въвеждане на данни за вход, след което започва командния цикъл. Класът също така има частни методи, които се извикват при въвеждането на конкретна команда.

```

1 class Program {
2 private:
3     static void addUserCommand(Library& library, bool isAdministrator);
4
5     static void changePasswordCommand(Library& library, const char*
        username);
6
7     static void addBookCommand(Library library, bool isAdministrator);
8

```

```

9      static void removeBookCommand(Library library, bool isAdministrator);
10
11     static void viewCommand(const Library& library);
12
13     static void sortCommand(Library library);
14
15     static void findCommand(const Library& library);
16
17     static void printCommand(const Library& library);
18 public:
19     static int run();
20 };

```

Методът run() се извиква при стартиране на цялата програма.

```

1 #include "include/program/Program.h"
2
3 int main() {
4     return Program::run();
5 }

```

3.2.5 Класове за грешки

В проекта има дефиниран един абстрактен базов клас за грешка. Той дефинира в себе си код на грешката, който може да е от един от 2 изборни вида: BookErrorCode или LibraryErrorCode.

```

1 #include <exception>
2
3 template<typename T>
4 class Exception : public std::exception {
5 public:
6     T errorCode;
7 public:
8     explicit Exception(T errorCode);
9
10     T getErrorCode() const;
11

```

```

12     virtual const char *getErrorMessage() const = 0;
13
14     const char *what() const noexcept override;
15 };
16
17 template<typename T>
18 Exception<T>::Exception(T errorCode) : errorCode(errorCode) {}
19
20 template<typename T>
21 T Exception<T>::getErrorCode() const {
22     return this->errorCode;
23 }
24
25 template<typename T>
26 const char *Exception<T>::what() const noexcept {
27     return this->getErrorMessage();
28 }

```

Грешката се имплементира от 2 класа: BookException и LibraryException, които дефинират съответствасието между код на грешката и съобщението и. Тези грешки се обработват от класа Program и съобщенията им се представят на потребителя, когато това е нужно.

```

1 BookException::BookException(BookErrorCode errorCode) :
    Exception(errorCode) {}
2
3 const char *BookException::getErrorMessage() const {
4     switch (this->errorCode) {
5         case CONTENTS_FILE_READING_ERR:
6             return "Book_content_file_could_not_be_opened_for_reading!";
7         case CONTENTS_FILE_WRITING_ERR:
8             return "Book_content_file_could_not_be_opened_for_writing!";
9         case CONTENTS_FILE_REMOVAL_ERR:
10            return "Book_content_file_could_not_be_deleted!";
11        case INVALID_RATING_RANGE:
12            return "Invalid_rating!_Rating_should_be_between_0.0_and_10.0.";
13        case INVALID_ISBN_LENGTH:
14            return "Invalid_ISBN_length!";

```

```

15     case INVALID_ISBN_GS1_PREFIX:
16         return "Invalid_ISBN_GS1_prefix!";
17     case INVALID_ISBN_CHARACTERS:
18         return "Invalid_characters_in_ISBN!";
19     case INVALID_ISBN_CHECKSUM_CHARACTER:
20         return "Invalid_ISBN_checksum_character!";
21     case INVALID_ISBN_CHECKSUM:
22         return "Invalid_ISBN_checksum!";
23     default:
24         throw std::invalid_argument("Unimplemented_item");
25 }
26 }

```

```

1 LibraryException::LibraryException(LibraryErrorCode errorCode) :
   Exception(errorCode) {}
2
3 const char *LibraryException::getErrorMessage() const {
4     switch (this->errorCode) {
5         case LibraryErrorCode::DUPLICATE_ISBN:
6             return "A_book_with_the_same_ISBN_already_exists!";
7         case LibraryErrorCode::DUPLICATE_USERNAME:
8             return "A_user_with_the_same_username_already_exists!";
9         case LibraryErrorCode::INVALID_OLD_PASSWORD:
10            return "Invalid_old_password!";
11        case LibraryErrorCode::MISMATCHING_PASSWORDS_ERR:
12            return "New_passwords_don't_match!";
13        case LibraryErrorCode::BOOK_NOT_FOUNT_ERR:
14            return "Book_not_found!";
15        default:
16            throw std::invalid_argument("Unimplemented_item");
17    }
18 }

```

3.2.6 Помощни класове

Програмата реализира два помощни класа. PasswordHelper предоставя методи за въвеждане на паролите и скриването им на стан-

дартния изход.

```
1  #ifdef _WIN32
2
3  #include <windows.h>
4
5  void PasswordHelper::enableInputEcho(bool on) {
6      // Get current console mode
7      DWORD mode;
8      HANDLE hConIn = GetStdHandle(STD_INPUT_HANDLE);
9      GetConsoleMode(hConIn, &mode);
10
11     // Change mode to enable/Disable echo for input characters and apply it
12     mode = on
13         ? (mode | ENABLE_ECHO_INPUT)
14         : (mode & ~(ENABLE_ECHO_INPUT));
15     SetConsoleMode(hConIn, mode);
16 }
17
18 #else
19
20 #include <termios.h>
21 #include <unistd.h>
22
23 void PasswordHelper::enableInputEcho(bool on) {
24     // Get current terminal settings
25     struct termios settings {};
26     tcgetattr(STDIN_FILENO, &settings);
27
28     // Enable/Disable echo for input characters and apply immediately
29     settings.c_lflag = on
30         ? (settings.c_lflag | ECHO)
31         : (settings.c_lflag & ~(ECHO));
32     tcsetattr(STDIN_FILENO, TCSANOW, &settings);
33 }
34
35 #endif
36
```

```

37 void PasswordHelper::inputPassword(char *passwordBuffer, unsigned int
    length, bool printNewLine) {
38     // Disable terminal character printing
39     enableInputEcho(false);
40
41     // Get password input
42     std::cin.getline(passwordBuffer, length);
43
44     // Enable terminal character printing
45     enableInputEcho(true);
46
47     // Print a new line after input
48     if (printNewLine) {
49         std::cout << "\n";
50     }
51 }

```

StringHelper предоставя един статичен метод за преобразуване на символни низове към символни низове единствено с малки букви.

```

1  const char *StringHelper::toLowerCase(const char *string) {
2      unsigned int length = std::strlen(string);
3      char *result = new char [length + 1];
4
5      for (unsigned int i = 0; i < length; i++) {
6          result[i] = (char) std::tolower(string[i]);
7      }
8      result[length] = '\0';
9
10     return result;
11 }

```

Глава 4.

Заключение