Курсов проект

БИБЛИОТЕКА

Автор:

Веселин Василев Живков

Факултетен номер: 72103

Телефон: +359888801246

E-mail: jivkovvesselin@gmail.com

Университет: СУ "Св. Климент Охридски"

Курс: 1

Група: 4

1. Цели и целева група

Линк към проекта в Github: https://github.com/GritzMaze/Library.git

1.1. Цели

- Представянето на информация на дадена книга по-удобен и лесен за четене начин
- Да поддържа база от данни със книги и потребители
- Да извършва операции за намиране, сортиране, добавяне, изтриване на книги и потребители
- Четене и записване на информацията във/от файл
- Изграждането на подходяща структура, която да поддържа наличните книги и потребителите, които да извършват операции с тях

1.2. Целева група

- Студенти
- Ученици
- Академични лица

2. Основни етапи:

- 2.1. Проучване на сложността на даденото задание
- 2.2. Определяне на основните елементи от структурата на програмата и техните функции
- 2.3. Планиране на графичното оформление и композиция
- 2.4. Избиране на методите, технологиите и библиотеките, чрез които да се постигнат поставените цели
- 2.5. Създаване на библиотеката
- 2.6. Събиране на информация за книги и попълване на базата данни
- 2.7. Тестване
- 2.8. Оценка на постигнатото
- При наличие на трудности при използването на програмата, тя ще бъде редактирана и тествана повторно
- 2.10. Финално публикуване

3. Стартиране на проекта

- 3.1. Изисквания за стартиране:
 - CMake
 - С++11 или по-висока версия на с++
 - C++ компилатор (i.e. MingW)
- 3.2. Стартиране на проект със CMake <u>How to Build a CMake-Based Project</u> (preshing.com)

4. Логическо и функционално описание

- 4.1. Програмата е с двуслойна архитектура:
 - 4.1.1. Презентационен слой чрез класът Draw данните от програмата се представят на крайния потребител
 - 4.1.2. Данните се съхраняват във файлове със разширения .books и .users. Книгите се съхраняват в .books, а потребителите в .users
- 4.2. Модули програмата е разделена на два модула представителен(потребителски) и административен
- 4.3. Функции на модулите
 - 4.3.1. *Представителен* представителният модул е видим за всички потребителите на програмата.
 - 4.3.2. *Административен* видим само за регистрирани потребители с администраторски права.

Чрез него се добавят/изтриват потребители в програмата и се запазва съдържанието на програмата.

5. Реализация

Използва се Обектно-ориентирано програмиране на C++ за реализацията на проекта.

За интерфейса са използвани биоблиотеките "windows" и "conio".

Работата със низове е постигната чрез ръчно написан String. Програмата използва и ръчно написан вектор.

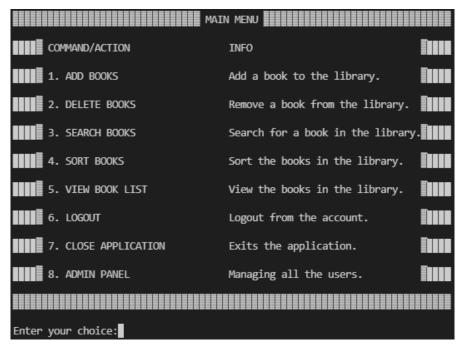
За управление на проекта и конструиране на проекта е използвано CMake.

6. Описание

6.1. Начален екран при стартиране на програмата:

6.2. Главно меню

- Добавяне на книга добавяне на книга в системата;
- Изтриване на книга изтриване на книга от системата;
- Търсене на книга Търсене на книга по име/автор/ключова дума;
- Сортиране на книга Сортира книгите по име/автор/година/рейтинг;
- Списък с книги Показва списък с наличните книги в системата;
- Bxod/M3xod Дава възможност на потребителя да влезе в акаунта си;
- Затваряне на програмата Затваря програмата;
- *Администраторски панел* Само потребителите с администраторски достъп виждат и могат да достъпят това меню;



6.3. потребителите в програмата, както и за запазването на съдържанието от програмата в файлове.

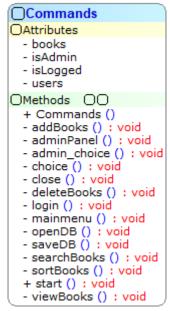
При стартирането на програмата за първи път, по подразбиране се създава потребител с потребителско име — \mathbf{admin} и парола — $\mathbf{i} < \mathbf{3c} + \mathbf{+}$. Потребителят е администратор.

- 6.3.1. Администраторско меню
 - Добавяне на потребител добавя потребител в системата
 - Изтриване на потребител изтрива потребител от системата
 - Списък с потребители извежда списък с писък с наличните потребители
 - *Отваряне на база данни* отваря база данни с книги и потребители
 - Запазване на база данни запазва базата данни на компютъра
 - Главно меню връща обратно към главното меню

ADMIN MENU		
COMMAND/ACTION	INFO	
1. ADD USER	Add a user.	
2. DELETE USER	Delete a user.	
3. VIEW USERS	View a list of users.	
4. OPEN DATABASE	Open a database.	
5. SAVE DATABASE	Save the database.	
6. MAIN MENU	Go to the main menu.	
Enter your choice:		

7. Описание на класовете

7.1. Клас *Commands* — Сърцето на програмата. Съдържа основната информация за това дали потребител е влязъл в системата и дали е администратор. Разпределя задачите по класове.



7.2. Клас *Draw* – Грижи се за графичната част от приложението. Чертае рамките, таблиците и менютата. Позиционира информацията в конзолата.

```
Draw
○Methods ○○
+ drawAdminMenu (): static void
+ drawFooter (const int &): static void
+ drawHeader (const String &): static void
+ drawMenu (const bool &, const bool &): static void
+ drawMenuElem (const String &, const String &, const int &): static void
+ drawMenuElem (const String &, const String &, const int &): static void
+ drawStart (): static void
+ drawSubmenu (const String &, const String &, const String &, const String &): static void
+ gotoxy (const int &, const int &): static void
+ pressAny (const int &): static void
```

7.3. Клас *Database* – Интерфейсен клас за базата данни от потребители и книги.

```
Database

○Methods ○○

+ Database ()

+ add (): virtual void

+ doesItExist (const String &): virtual bool

+ open (const String &): virtual void

+ remove (const size_t &): virtual bool

+ save (const String &): virtual void

# view (): virtual void

+ ~Database (): virtual
```

7.4. Клас *BookDB* – Наследява Database. Грижи за запазените книги в системата и тяхната обработка.

```
□BookDB
OAttributes

    books

OMethods OO
 + BookDB ()
 + add () : void
 + deleteBooks (): void
 + doesItExist (const String &) : bool
 + open (const String &): void
 + printByID (const unsigned int &) : const bool
 + remove (const size_t &) : bool
 + save (const String &) : void
 + search () : void
 - searchByAuthor (const String &) : int
 - searchByByKeyword (const String &): int
 - searchByTitle (const String &): int
 - searchChoice (): void
  - sort (const String &, const String &) : void
 + sortBooks (): void
 - sortBy (const String &) : void
 - sortByAuthor (const String &): void
 - sortByRating (const String &): void
 - sortByTitle (const String &): void
 - sortByYear (const String &): void
 - sortChoice (): void

    sortChoice2 (const String &): void

 - sortCross (const String &, const String &): void
 - view (): void
  + viewBooks (): void

    viewSearched (const String &): void

 + ~BookDB ()
```

7.5. Клас *UserDB* – Наследява Database. Грижи се за запазените потребители в системата и тяхната обработка.

```
UserDB

OAttributes
- users

OMethods OO
+ UserDB ()
+ add (): void
+ addUser (): void
+ deleteUser (): void
+ doesItExist (const String &): bool
+ login (bool &, bool &): void
+ open (const String &): void
+ remove (const size_t &): bool
+ save (const String &): void
- view (): void
+ viewUsers (): void
+ ~UserDB ()
```

7.6. Клас *Book* – Описва дадена книга. Грижи се за обработката на единична книга.

```
Book
 ○Attributes
     - _id
- author
     - description
    - genre
- id
     - keywords
     - rating
     - title
     - yearOfPublish
OMethods OO
    + = (const Book &): Book &
+ Book ()
+ Book (const String &, const String &, const String &, const size_t &, const Vector < String > &, const double &)
+ Book (const char *, const char *, const char *, const size_t &, const Vector < String > &, const double &)
+ getAuthor (): const String &
+ getPage (): const String &
+ getPage (): const String &
     + getDesc () : const String &
+ getGenre () : const String &
     + getID () : const size_t &
+ getKeywords () : const Vector < String > &
     + getReywords () : const vector < string >
+ getRating () : const String &
+ getTitle () : const String &
+ getYOP () : const size_t &
+ open (const std :: fstream &) : const void
     + partlyPrint () : const void
+ partlyPrintVertical (const int &) : const void
     + print () : const void

+ print () : const void

+ save (std :: ofstream &) : const void

+ setAuthor (const String &) : void

+ setDesc (const String &) : void

+ setDescFromInput () : void

+ setDescFromInput () : void
    + setDescFromInput (): void

+ setGenre (const String &): void

+ setGenreFromInput (): void

+ setKeyWords (const Vector < String > &): void

+ setKeyWordsFromInput (): void

+ setKeyWordsFromString (const String &): void

+ setRating (const double &): void

+ setTitle (const String &): void

+ setTitleFromInput (): void

+ setTitleFromInput (): void

+ setTitleFromInput (): void

+ setYOP (const size t &): void
      + setYOP (const size t &) : void
      + setYOPFromInput (): void
```

7.7. Клас *User* – Описва даден потребител. Грижи се за обработката на единичен потребител.

```
User
OAttributes
   - _id
  - admin
   - id
   - password
   - username
OMethods OO
   + != (User *) : bool
+ != (const User &) : bool
   + = (const User &) : User &
+ == (User *) : bool
  + == (const User &) : bool

+ User ()

+ User (const String &)

+ User (const User &)

+ User (const String &, const String &, bool)

- copy (const User &) : void

+ getID () : const size_t

+ getPassword () : const String

+ getUsername () : const String

+ isAdmin () : bool

+ load (std :: ifstream &) : void

+ print () : void
   + == (const User &) : bool
    + print () : void
    + printVertical (const int &) : const void
    + save (std :: ofstream &) : const void
    + setAdmin (bool) : void
    + setPass (const String &) : void
    + setUsername (const String &): void
```

7.8. Клас *InputHandle* – Занимава се с извеждането на грешките в конзолата по добър за четене начин.

7.9. Клас *String* – Използва се за представяне на информацията във формата на текстов вид.

```
String
OAttributes

    capacity

    data

    size

OMethods OO
  + != (const String &) : bool
  + != (const char *) : bool
  + + (const String &) : String
  + + (const char *) : String
+ + (const char &) : String
  + += (const String &) : String &
  + += (const char *) : String &
  + += (const char &) : String &
  + < (const String &) : bool
  + = (const String &) : String &
  + = (const char *) : String &
  + = (const Vector < char > &) : String &
  + == (const String &) : bool
  + == (const char *) : bool
  + == (const char &) : bool
  + > (const String &) : bool
  + String ()
  + String (const String &)
  + String (const char *)
  + [] (int &) : char &
  + [] (const int &) : const char &
  + add (const char &) : void
  - copy (const char *) : void
  - create (const size_t &) : char *
  - erase (): void
  + findElem (const char &) : bool
  + getCapacity () : const int
  + getLength () : const int
  + getString () : const char *
  + inputProtected () : String
  + insertAt (const char &, int) : void
  + print () : const void
  + removeAt (int &) : void
  resize () : void
  + setCapacity (const int) : void
  + setSize (const int) : void
  + setString (const char *) : void
  + trimEnd () : void
+ trimEnd (int &) : void
  + trimStart () : void
  + trimStart (int) : void
  + ~String ()
```

7.10. Клас *Vector* – Едномерен масив за съхранение на информация.

```
○Attributes
  - arr
  - capacity

    size

OMethods OO
  + = (const Vector < T > &) : Vector < T > &
  + Vector ()
+ Vector (const T *, const size_t &, const size_t &)
  + [] (int &) : T &
  + [] (const int &) : const T &
  + back () : const T &
  - copy (const Vector < T > &) : void
  - erase () : void
  + front (): const T &
  + getElem (const size_t &) : const T
  + getSize () : const size_t
+ getcapacity () : const size_t
  + isExist (const T &) : bool
  + isExistUser (const T &) : bool
  + popBack () : void
  + print () : const void
  + pushBack (const T &) : void
+ pushFront (const T &) : void
  + remove (const size_t &) : void
  - resize () : void
  + swap (T &, T &) : void
  + ~Vector ()
```