***Тема 2: Електронна библиотека***

*Георги Николаев Атанасов, ФН: 3MI0800092, I курс, I група*

*Github repo: https://github.com/nigosto/library-system*

***Описание и идея на проекта:***

Основната идея на проекта е да се реализира електронна библиотека, която да съхранява информация за книгите в нея и при нужда тази информация да се предоставя на потребителя или да бъде променяна. Освен това чрез приложението трябва да могат да се четат книгите от библиотеката във формат удобен на потребителя.

***Цел и задачи на разработката:***

Целта на проекта е да се реализира необходимата функционалност, така че да може да се осъществи правилна и лесна работа в диалогов режим с библиотеката. Главните задачи са да се съхранява добре информацията за библиотеката във файлове, да се изгради добра структура на проекта, основана на добрите ООП принципи, и да се обработват по най-добър и правилен начин командите на потребителя. Освен това трябва да се имплементират две нива на достъп – неоторизиран и оторизиран, като трябва да се вземат всички мерки свързани със сигурността и разделението на функциите на двата вида потребители.

***Обща архитектура:***

Както бе споменато по-горе основната задача, свързана с изграждането на архитектурата, е да се спазят всички ООП принципи. За целта проектът е разделен на класове, като всеки клас отговаря за определена част от изпълнението на задачата. Освен това е имплементиран един основен клас, който да разпределя работата измежду останалите, като идеята зад него е, че по този начин програмата може лесно да се превърне в библиотека и да се използва и в други проекти, които разчитат само на ограничен брой функции. Всеки клас отговаря за коректността на своите данни, като по този начин се избягва излишната валидация във функциите, които използват съответните класове. Освен това всеки клас е собственик на паметта, т.е. всеки клас отговаря за инициализирането на член-данните и при приключване на жизнения му цикъл - за освобождаването на паметта, заделена при инициализация. С една дума спазен е принципа *RAII – Resource Acquisition is Initialization.*

***Реализация:***

Програмата е разделена на 3 класа: *Book, Library* и *LibraryHandler*.

1. *Book* - представлява абстракция на книга с член-данни: title, author, filename, description, rating и ISBN. Инвариантата на класа е: ISBN трябва да се състои точно от 13 символа, rating да бъде в интервала от 1 до 5 и filename да бъде коректно има на текстов файл. Освен това title, author, filename, description и ISBN са статични масиви от char, защото информацията за книгите се пази в двоични файлове. Въпреки това размерностите на тези масиви са достатъчно големи, за да поберат повечето от подадените им данни и ако се опита да се присвои стойност с по-голям размер, то тя най-вероятно няма да е коректна (например би било нелогично да има книга със заглавие съдържащо повече от 255 символа), и затова има валидация, която изрично проверява за коректни размери на подадените данни. Съответно няма реализирана голяма четворка за този клас. Освен стандартните селектори за достъп и мутатори за промяна на член-даниите са имплементирани и функциите showTextBySentences() и showTextByPages(size\_t rows), които изкарват текста на книгата на екрана, съответно по изречения и по страници, като изрично се казва по колко реда да има на страница.
2. *Library* – представлява абстракция на библиотека, която е реализирана като колекция (масив) от книги. Понеже този път масивът от книги е динамичен, има реализирана голяма четворка. Освен това са реализирани основни функции, взаимствани от тези на std::vector: resize(), size(), empty(), insert(const Book& book), remove(const char\* \_isbn, bool removeFile = false), at(size\_t position) и operator[](size\_t position). Единствената от тях, която се различава по функционалност от съответната функция от std::vector е remove, която приема допълнителен параметър оказващ дали да се изтрие текстовият файл на книгата заедно с обекта в масива. Освен това има реализирани функции за сортиране: sortByTitle(), sortByAuthor() и sortByRating(), които сортират библиотеката съответно по заглавие, автор и рейтинг, както и за намиране на книга: findByTitle(const char\* \_title), findByAuthor(const char\* \_author), findByISBN(const char\* \_isbn) и findByDescription(const char\* \_description), които съответно намират кнога по нейно заглавие, автор, уникален номер или част от описанието. Последните функции не гарантират за намирането на дадена книга и затова те се различават от други, подобни на тях функции, по това че връщат указател, а не референция (в случая, когато не бъде намерена връщат nullptr).
3. *LibraryHandler* – представлява основният клас на програмата, който обработва потребителските данни и обединява работата на останалите класове. Като член-дании той има библиотека и булева променлива, която показва дали потребителя, който в момента ползва прогрмата, е с оторизиран достъп. Реализирани са функциите readLibrary() и saveLibrary(), които отговарят за това да прочетат книгите от двоичния файл и да ги запазят в библиотеката и след приключване на работата да ги запишат обратно в същия файл. Функцията authenticate() отговаря за автентикацията на потребителя, ако той е администратор. Първо тя извежда въпрос дали потребителя е администратор и при отговор *yes* чака за подаване на вярна парола. При писане, паролата е скрита и на нейно място се появяват „\*“, като тази функционалност е реализирана чрез функцията getch() от библиотеката <conio.h>, която е САМО ЗА *WINDOWS*. Функцията readCommands() отговаря за това да обработи командите на потребителя, като за тази цел тя използва помощните функции handleListAll(), handleFind(), handleAdd(), handleRemove() и handleShowText(). За по-лесна работа между конзолата и потребителя, горната функция извежда подходящи съобщения на екрана и очаква отговора на потребителя, като се грижи за валидацията на отговорите.

Във функцията main() се използва единствено класът *LibraryHandler*, където се извикват само функциите, необходими за правилната работа на програмата. По този начин се постига добра четимост на основния файл на програмата.

***Обобщение на изпълнението на началните цели:***

Във връзка с това, че цялата функционалност от изискванията е реализирана, програмата обработва без проблем командите на потребителя, и то по доста интуитивен начин, и това че класовете работят добре помежду си, смея да твърдя, че началните цели на проекта са изпълнени. Освен това информацията за книгите се пази в двоичен файл, което предотвратява лесната ѝ промяна от човек с достъп до него. Не на последно място, реализирани са голям брой валидации за входните данни, които да предотвратят нежелано поведение на програмата, като при некоректни данни се извеждат подходящи съобщения, които да информират по правилен начин потребителя.

***Насоки за бъдещо развитие и усъвършенстване:***

На първо място за усъвършенстване на проекта бих поставил това, той да работи и на други платформи освен *Windows*, понеже в момента логиката за паролата използва библиотеката <conio.h>, която е само за *Windows*. Друга насока за подобряване на приложението е всеки потребител да се регистрира при първото си влизане и след това да влиза в акаунта си, където да има персонална информация, като например любими книги, книги които в момента чете и т.н. Това естествено води до нуждата от реализиране на повече функционалности на програмата. Ще завърша с това, че в бъдеще би могъл да се реализира потребителски интерфейс, така че приложението да бъде използвано много по-лесно и интуитивно, отколкото е в момента с конзолата.