Библиотека

(Документация)

Глава 1. Увод

* 1. Описание и идея на проекта

Проектът Библиотека има за цел да реализира информационна система, която поддържа библиотека, като програмата трябва да съхранява и обработва данни за наличните книги във файл. Системата поддържа два вида потребители- администратор и обикновен потребител, като в началото при стартиране на програмата няма налични книги и има само един потребител, който е администратор, с потребителско име admin и парола i<3c++.

* 1. Цел и задачи на разработката

Основната цел на проекта е да бъде разработена програма, която да поддържа база данни от книги, които са част от билиотеката и списък с потребители и администратори.

За реализацията на този проект е нужно данните за книгите и потребителите да бъде представена по максимално най-добрия начин. За целта информацията за всяка една книга ще бъде представена в отделен клас, който има член-данни за автора на книгата, заглавието, жанр, кратко описание, година на издаване, ключови думи, рейтинг и уникален номер за библиотеката. Тъй като в заданието на самият проект се изисква да се търси книга по ключова дума, за това ще бъде реализиран клас, който да харектеризира ключова дума.

Информацията за всеки потребител също ще бъде представена в отделен клас с член-данни за потребителското име и парола. Указано, че за да се изпълни дадена функция може да се изисква да има влязъл потребител, или пък да няма такъв, или пък е нужно да има адниминастатор (admin), или пък да няма никакви изисквания да има влязъл потребител или наличие на администрантор. Поради тази причина няма да бъде реализиран отделен клас за администратор. Проблемът с изпълнението на всяка функция ще бъде решен, като по време на изпълнение на програма се проверява дали има влязъл потребител.

За библиотеката ще бъде реализиран отделен клас, който ще съдържа списък с книги, техният брой и капацитет.

За стартирането на програмата ще бъде нужен още един клас, който да поддържа една библиотека и всички функции за работа с нея.

Глава 2. Преглед на предметната област

2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

За реализирането на проекта е необходимо да бъдат дефинирани класове, които са основен инструмент на ООП и са средство за дефиниране на абстракти типове данни. За описанието на това, какво са потребител, книга, библиотека и ключова дума, ще бъдат използвани класове, които ще бъдат свързани помежду си.

За сортирането на книгите по даден критерии- година на издаване и рейтинг, ще бъде използван метода на мехурчето за сортиране във възходящ и низходящ ред на числа. Аналогично с помощта на функцията strcmp, която е част от библиотеката cstring, ще бъдат сортирани книгите по заглавие и име на автор.

2.2. Дефиниране на проблем и сложност на поставената задача

Тъй като книга може да бъде търсена по ключова дума, няма как ключовите думи да бъдат представени като низ от думи, разделени със запетаи.

Книгите в библиотеката може да бъдат сортирани по два начина- във възходящ и низходящ ред, и по четири критерия- по име на автор, заглавие, година на издаване и рейтинг. Това означава, че трябва да бъдат реализарини различини методи за сортиране на числа и за сортиране на низове.

Част от функциите, които се изискват се изпълняват при наличието на вписан потребител и администратор, или на само вписан потребител без да се изисква наличието на администратор, или въобще да няма нужда нито от потребител, нито от администратор.

2.3. Подходи, методи за решаването на поставените проблеми

За да може да бъде реализирана функция, която да търси книга по ключова дума, ще бъде реализиран отделен клас, който ще характеризира една ключва дума. Всяка книга ще има списък (масив от указатели) към обекти на този клас, брой на ключовите думи и капацитет. Така посредством селектора на класа, за ключовите думи, ще има достъп до всяка една от тях.

За всеки един от критериите за сортиране ще бъде създадена функция в класа за библиотеката, която да сортира във възходящ и низходящ ред книгите. Сортирането по година на издаване и рейтинг ще бъде направено чрез стандартния метод на мехурчето. Сортирането на книгите по заглавие и автор ще бъде реализирано отново чрез гореспоменатия метод и с помощта на помощта на функцията strcmp, която е част от библиотеката cstring. С тази функция ще бъдат сверявани заглавията и имената на авторите на книгите.

За да разберем дали може да се изпълни някоя от функциите, които са описани във файла с проектите по ООП, ще бъде използвана помощна булева променлива, която ще е със стойност истина, ако има успешно влязъл потребител, и стойност лъжа в противен случай и още една булева променлива, която има стайност истина, ако успешно е прочетен файл с книги, и лъжа в противен случай. Всеки път преди изпълнението на въведена функция ще се проверява дали има успешно отворен и прочетен файл и дали са спазени изискания за наличие на вписан потребител или администратор в системата.

2.4. Потребителски и качествени изисквания

За да бъдат изпълнени всички функции за дадена библиотека е необходимо винаги да има успешно отворен и прочетен файл с книги и да са изпълнени правата за всяка една функция. След това е програмата очаква да се въведе команда, като след въвеждането се изпълнява според дефинираните правила. Това действие ще продължава до въвеждането на командата “exit”.

Глава 3. Проектиране

3.1. Обща архитектура- ООП дизайн

Първоначално трябва да бъде реализиран касът, който характеризира потребител. Той притежава член-данни за потребителско име и парола, които са от тип char\*. За класа е реализирана голяма четворка, която да се погрижи за правилното създаване на обекти от този клас. Като помощни функиции са реализирани методи за копиране (void copyFrom(const User&)) на член-данни и за освобождаване на заделената динамична памет в конструкторите (void cleanMemory()). Класът User има и релаизарини селектори и мутатори. Тъй като информацията трябва да бъде записана във файл- база данни с потребители, класът има реализирани методи за записване и четене от бинарен файл.

За създаването на клас, който да описва книга, е необходимо да бъде реализиран помощен клас за ключова дума (class Tag). Той има само една член-данна за думата, от тип char\* . Класът има реализирана голяма четворка, селектор, мутатор и методи за записване и четене от бинарен файл, който ще бъдат извикани при записването и прочитането на книга от бинарен файл.

Информацията за книга също е представена в отделен клас Book. Той има член-данни за името на автора, заглавие, жанр, описание, списък с ключови думи(Tag\* ), брой и капацитет на ключовите думи, рейтинг и идентификационен номер на книгата. Класът има реализирана голяма четворка и две помощни функции, които да копират член- данните и да изтриват заделената за тях динамична памет (void copyFrom(const Book&); void cleanMemory();). Също така има дефинирани селектори и мутатори. Тъй като всяка книга трябва да бъде записана във файл са нужни два метода, които да прочитат и записват информацията на книга във даден бинарен файл. Класът притежава и методи за извеждане на пълна и частична информация за книга, както и предефиниран оператор за сравнение на две книги, операторът враща истина, ако две книги са с едно и също име, в противен случай връща лъжа.

Всяка библиотека представлява списък от книги, за това реализиран отделен клас за библиотека (class Library). Той претежава член-данни за брой книги, капацитет на библиотеката (private член-данни). Класът има реализирани голяма четворка, функция за копиране на член-данни (void copyFrom(const Library&);), за освобождаване на заделената памет за член-данните (void cleanMemory();) и преоразмераване на масива от книги (void resizeLib();). Останалите методи на класа са:

//Добавяне на нова книга.

void addBook(const Book&);

//Изтриване на книга.

void removeBook(const Book&);

//Извеждане на информацията за всяка книга от библиотеката.

void booksAll();

//Извеждане на подробна информация за книга с персонален номер

void booksInfo(unsigned);

//Търсене на книга по заглавие.

void booksFindTitile(char\* title);

//Търсене на книга по автор. Типът на функцията е void, защото може да има няколко различни книги от един автор.

void booksFindAuthor(char\* author);

//Търсене на книга по ключова дума. Типът на функцията е void, защото може да има няколко книги с такава ключова дума.

void booksFindTag(char\* tag);

//Сортиране на книгите във възходящ ред по година на производство.

void sortBooksOfYearOfIssueAsc();

//Сортиране на книгите в низходящ ред по година на производство.

void sortBooksOfYearOfIssueDesc();

//Сортиране на книгите във възходящ ред по рейтинг.

void sortBooksOfRatingAsc();

//Сортиране на книгите в низходящ ред по рейтинг.

void sortBooksOfRatingDesc();

//Сортиране на книгите във възходящ ред по заглавие.

void sortBooksOfTitleAsc();

//Сортиране на книгите в низходящ ред по заглавие.

void sortBooksOfTitleDesc();

//Сортиране на книгите във възходящ ред по име на автора.

void sortBooksOfAuthorAsc();

//Сортиране на книгите в низходящ ред по име на автора.

void sortBooksOfAuthorDesc();

//void runProgram();

void writeBooksToFile( std::ofstream&);

void readBooksFromFile(std::ifstream&);

За стартирането на програмата е реализиран отделен клас Manager. Той има само една член-данна от тип Library. Класът притежава една публична член-данна, която да пази името на последният отворен файл, така че командата save() да се изпълни за последно отворния файл. Дефинирани са функции за отваряне на файл, затваряне на файл, запазване на текущия отворен файл, запазване на информацията за книгите от библиотеката в нов файл, помощна функция, функция за вписване на потребител, за отписване на потребител, за изход от програмата, за извеждане на информация за книга с даден идентификационен номер, за търсена на книга по даден критерий- заглавие автор и ключова дума, за добавяне на нов потребител, за премахване на нов потребител, за добавяне на книга, за премахване на книга, сортиране на книгите и функция за стартиране на програмата. Декларациите на функциите са:

void open(char \*);

void closeFile();

void save();

void saveAs(char\*);

void help() const;

void login();

void exit();

void logout();

void booksAll();

//Извеждане на информацията за книга с даден номер.

void booksInfo(unsigned ID);

//Търсене на книга по определен критерии и стойност на критерия.

void booksFind(char\* option, char\* optionString);

//Добавяне на потребител.

void userAdd(char\* username, char\* password);

//Премахване на потребител.

void userRemove(char\* username);

//Прамахване на книга.

void booksRemove(const Book&);

//Сортиране на книгите по даден критерии. По подразбиране сортирането ще е във възходящ ред.

void booksSort(char\* option, char\*type = "asc");

//Добавяне на книга.

void booksAdd(const Book&);

//Стартиране на програмата.

void runProgram();

Глава 4: Реализация, тестване

4.1. Реализация на класове

Тъй като една книга може да бъде търсена по три критерия- заглавие, автор и ключова дума, са създадени за класа Library три метода, които да откриват книга по един от трите критерия.

void Library::booksFindTitile(char \* title)

{

bool isFind = false;

for (int i = 0; i < m\_countBook; ++i) {

if (strcmp(title, m\_books[i].getTitle()) == 0) {

isFind = true;

//Извеждане на информацията за книгата

m\_books[i].print();

std::cout << std::endl;

}

}

//Извеждане на съобщение, ако не е намерена книга

if (isFind == false) {

std::cout << "The book entitled \"" << title << "\" was not found." << std::endl;

}

}

void Library::booksFindAuthor(char \* author)

{

bool isFind = false;

for (int i = 0; i < m\_countBook; ++i) {

if (strcmp(author, m\_books[i].getName()) == 0) {

isFind = true;

//Извеждане на информацията за намерената книга.

m\_books[i].print();

std::cout << std::endl;

}

}

//Извеждане на съобщение, ако не е намерена книга.

if (isFind == false) {

std::cout << "There are no books by the author " << author << " in the library." << std::endl;

}

}

void Library::booksFindTag(char \* tag)

{

bool isFind = false;

for (int i = 0; i < m\_countBook; ++i) {

for (int j = 0; j < m\_books[i].getNumberTag(); ++i) {

Tag\* tags = new(std::nothrow) Tag[m\_books[i].getNumberTag()];

if (tags == nullptr) {

std::cout << "Not enought memory for tags in booksFindTag. Error!" << std::endl;

return;

}

tags = m\_books[i].getKeywords();

//Преминаваме през всички ключови думи на поредната книга и ги сверяваме дали са като tag.

if (strcmp(tag, tags[j].getTag()) == 0) {

isFind = true;

m\_books[i].print();

break;

}

}

}

//Ако не е намерена книга с исканата ключова дума.

if (isFind == false) {

std::cout << "No books found." << std::endl;

}

}

В класа, който стартира програмата, когато потребителят иска да потърси книга в библиотеката по даден критерии, в зависимост от това какъв критерии е въвевел потребителя ще бъде извикана една от горноописаните три функции за библиотеката.

За сортирането на книгите в библиетеката са реализирани отделни функции за сортиране по определен критирии и вид на сортиране в класа Library. Аналогично на търсенето на книга, и при сортирането ще бъде извикана съответаната функция за сортиране, след като потребителият е въвел коректни данни за критерия за сортиране и типа на сортировката.

Пример за сортиране на книгите по година на издаване във възходящ ред.

void Library::sortBooksOfYearOfIssueAsc()

{

for (int i = 0; i <m\_countBook; ++i) {

for (int j = 0; j < m\_countBook - i - 1; ++j) {

if (m\_books[j].getYearOfIssue() > m\_books[j + 1].getYearOfIssue()) {

Book tempBook = m\_books[j];

m\_books[j] = m\_books[j + 1];

m\_books[j + 1] = tempBook;

}

}

}

}

Пример за соритиране на книгите по заглавие в низходящ ред.

void Library::sortBooksOfTitleDesc()

{

for (int i = 0; i < m\_countBook; ++i) {

for (int j = i + 1; j < m\_countBook; ++j) {

if (strcmp(m\_books[i].getTitle(), m\_books[j].getTitle()) < 0) {

Book tempBook;

tempBook = m\_books[i];

m\_books[i] = m\_books[j];

m\_books[j] = tempBook;

}

}

}

}

4.2. Управление на паметта и алгоритми. Оптимизации

В почти всички от реализираните класове се заделя динамична памет в конструкторите. За да няма повтраяне на код, всеки от класовете притежава функции, които заделят памет и освобождават, заделетана памет, когато е нужно. Такива функции за класа Book например са:

void copyFrom(const Book&);

void cleanMemory();

Аналогични на тези функции, притежават и остналите класове.

Класове, които притежават масиви от указатели към други обекти на класове, притежават функция, която преоразмерява този масив,когато е дастигнат капацитета му. Обикновено преорзамеряването става при добавянето на нов обект към масива.

Пример за такава функция:

void Library::resizeLib()

{

unsigned newCapacity = m\_capacity \* 2;

Book\* newBooks = new (std::nothrow) Book[newCapacity];

if (newBooks == nullptr) {

std::cout << "Not enought memory for resize! Error!" << std::endl;

return;

}

for (unsigned i = 0; i < m\_capacity; ++i) {

newBooks[i] = m\_books[i];

}

m\_capacity = newCapacity;

delete[] m\_books;

m\_books = newBooks;

}

void Library::addBook(const Book & newBook)

{

if (m\_countBook >= m\_capacity) {

resizeLib();

}

m\_books[m\_countBook] = newBook;

++m\_countBook;

}

При създаването на класовете Library, Manager, Book бяха необходими константи за инициализирането на капацитет или на константен низ. Ако бяха дефинирани като глабални константи за цялата програма, то е възможно тези константи да бъдат промени и да се наруши смисъла и реализацията на програмата. За това, когато са необходими константи за създаването на обект от даден клас, то в този клас константите са статични и по този начин те са „глобални“ за класа, но няма външен дастъп до тях и не могат да бъдат променени.

При създаването на всяка книга, трьбва да се зададе нейн уникален идентификационен номер. За целта е използвана статична променлива (static unsigned m\_ID;), която се увеличава с единица при всяко създване на книга. Член-данната, която служи за иденитификационен номер на книга (unsigned m\_bookID;) присвоява стойността на статичната променлива (m\_ID) при създаването на нова книга (в конструктора) и след това се инкрементира с едница. m\_ID е статична променлива, която е част от статичната памет, и така след изпълнение на конструкторите тя ще бъде променена.

4.3. Планиране, описание и създаване на тестови сценарии

По време на създаване на класовете в програмата бяха създадени тестове, които да проверят дали правилно се създават обекти от различните класове, дали методите им работят правилно, дали е направена пълна валидация на данните, които постъпват преди да се създаде обект от даден клас. Всички тестове се намаират в заглавен файл с име “Tests.hpp”.

За класа User е важно да бъдат тествани дали правилно работят конструкторите, мутаторите, селекторите, функцията за записване на потребител във бинарен файл и функцията за четене от бинарен файл.

void testUserClass() {

User u1;

u1.setUsername("ani123");

u1.setPassword("123456789987654321");

std::cout << "Username: " << u1.getUsername() << std::endl

<< "Password: " << u1.getPassword() << std::endl;

User u2 = u1;

std::cout << std::endl;

std::cout << " User 2 username: " << u2.getUsername() << std::endl

<< "User 2 password: " << u2.getPassword() << std::endl;

User u3("User", "MyPassword123");

std::cout << "User 3 username: " << u3.getUsername() << std::endl

<< "User 3 password: " << u3.getPassword() << std::endl;

User u4;

u4 = u3;

std::cout << "User 4 username: " << u4.getUsername() << std::endl

<< "User 4 password: " << u4.getPassword() << std::endl;

}

Аналогично за класа Tag трябва да бъдат направени същите тестове, като от основно значение са методите за сериализация и десериализация.

void testTagClass() {

Tag tag("super hero");

std::cout << "Tag: " << tag.getTag() << std::endl;

std::ofstream ofs("Tag.bin", std::ios::out | std::ios::binary);

tag.serializeTag(ofs);

ofs.close();

std::ifstream ifs("Tag.bin", std::ios::in | std::ios::binary);

Tag tag1;

tag1.deserializeTag(ifs);

ifs.close();

std::cout << "Tag2: " << tag.getTag() << std::endl;

}

За класа Book трябlа да бъдат отново тествани конструкторите и методите за четене и писане в бинарен файл, мутаторите, селекторите и предефинираният оператор за сравнение.

Аналогично на другите класове, и за класа Library трабва да се тестват конструкторите и методите за четене и писане в бинарен файл, мутаторите, селекторите, всички сортировки и функциите за намиране на книго по определен критерии.

Класът, който стартира програмата е тестван по време на изпълнение с подадени от потребителя данни.

Глава 5. Заключение

5.1. Обобщение на изпълнението на началните цели

Началните цели на проекта са изпълнени, използвайки знанията от курса по УП и ООП.

5.2. Насоки за бъдещо развитие и усъвършенстване

В бъдеще проектът може да бъде усъвършенстван. Възможно е администраторът да придобие други правомощия и това да доведе до създаване на клас за администратор и по време на изпълнение на програмата да се проверява дали вписаният потребител е администратор или обикновен потребител. Възможно е при въвеждането на парола на екрана да се изписва някакъв символ, за да остане паролата скрита. А сащо така при регистрация на нов потребител да се проверява сложността на паролата- дали има малко и големи букви, символи и цифри.

Линк към Github хранилище: <https://github.com/TerezaTrenadafilova/Library-project>