МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ»)

Факультет экономики, менеджмента и информационных технологий
(факультет)

Кафедра Систем управления и информационных технологий в строительстве

КУРСОВОЙ ПРОЕКТ

по дисциплине Объектно-ориентированное программирование

тема Разработка программного обеспечения с использованием объектно-ориентированного подхода

Расчетно-пояснительная записка

Разработал студент _____ 26.05.22 О.В. Артемова
Подпись, дата Инициалы, фамилия

Руководитель _____ Е.Н. Королев
Подпись, дата Инициалы, фамилия

Члены комиссии _____
Подпись, дата Инициалы, фамилия

_____
Подпись, дата Инициалы, фамилия

Нормоконтролер _____ Е.Н. Королев
Подпись, дата Инициалы, фамилия

Защищена 26.05.22 Оценка отл
дата

2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ВГТУ»)

Кафедра Систем управления и информационных технологий в строительстве

ЗАДАНИЕ
на курсовой проект

по дисциплине Объектно-ориентированное программирование

тема Разработка программного обеспечения с использованием объектно-ориентированного подхода

Студент группы ИСТ-214 Артемова Ольга Владимировна
Фамилия, имя, отчество

Вариант 1. Библиотека (Электронная библиотека)

Технические условия 11th Gen Intel(R) Core(TM) i3-1115G4 @ 3.00GHz 3.00 GHz, операционная система Windows 10, ОЗУ 8 ГБ

Содержание и объем проекта (графические работы, расчеты и прочее):
анализ предметной области и требований к программному обеспечению(7 страниц); моделирование и разработка системы на основе принципов ООП(11 страниц); реализация системы на общесистемном языке программирования(17 страниц); 30 рисунков, 3 таблицы, 1 приложение

Сроки выполнения этапов анализ предметной области и требований к программному обеспечению (01.03 – 25.03); моделирование и разработка системы на основе принципов ООП (26.03 – 19.04); реализация системы на общесистемном языке программирования (20.04 – 11.05); описание диалога с пользователем (12.05 – 21.05); оформление пояснительной записки (22.05 – 04.06)

Срок защиты курсового проекта 05.06.2022

Руководитель _____ Е.Н. Королев
Подпись, дата Инициалы, фамилия

Задание принял студент _____ О.В. Артемова
Подпись, дата Инициалы, фамилия

Содержание

Введение 4
1 Анализ предметной области 5
1.1 Особенности предметной области 6
1.2 Проблемы, возникающие в данной предметной области и перспективы их решения с использованием программных средств 8
1.3 Анализ существующих аналогов 9
1.4 Цель и задачи курсового проектирования 10
2 Моделирование и разработка системы на основе принципов ООП 12
2.1 Постановка задачи 12
2.2 Объектно-ориентированные принципы и компоненты 15
2.3 Разработка классов 17
3. Реализация системы на общесистемном языке программирования 23
3.1. Выбор средств программной реализации 23
3.2 Алгоритм выполнения программы 24
3.3 Модульная структура программы 24
3.4 Описание диалога с пользователем 26
Заключение 40
Список использованной литературы 41
Приложение 42

Введение

В настоящее время в связи с развитием компьютерной техники появилась возможность автоматизировать многие процессы, увеличился объем обрабатываемой информации, вследствие чего возникла объективная необходимость автоматизировать большую часть сферы человеческой деятельности.

Для принятия обоснованных и эффективных решений в производственной деятельности появилась возможность при помощи компьютеров и средств связи получать, накапливать, хранить и обрабатывать данные.

В реальной жизни в библиотеке взаимодействие человек-книга происходит не непрямую, а косвенно. Чтобы получить книгу, он должен обратиться к библиотекарю. Соответственно, книга проходит через руки библиотекаря, а тот, в свою очередь, уже лично передает ее читателю. В онлайн-библиотеке пользователь выбирает книги сам, не нуждаясь в помощи стороннего лица.

В рамках данной курсовой работы будет разработан соответствующий интерфейс для взаимодействия пользователя и библиотеки.

1 Анализ предметной области

В наши дни сервис электронных библиотек пользуется популярностью среди людей, которым привычнее читать книги в электронном формате, не выходя из дома.

Электронная библиотека представляет собой базу данных, содержащую издания и произведений разных литературных направлений и жанров различных многочисленных писателей и поэтов.

Поскольку поддержка данного сервиса достаточно трудоемка, бесплатный доступ к книгам бывает ограничен, и пользователи за денежную плату могут взять электронные версии книг в аренду на некоторое время или купить их за более высокую цену насовсем.

Задача данного сервиса заключается в предоставлении пользователям постоянного доступа к книгам без ограничения по времени или непостоянного доступа на некоторое время, по истечении которого доступ у пользователей будет отозван.

Для успешного и эффективного функционирования практически любой системы необходимы ведение контроля, анализа и внедрение автоматизации для отдельных процессов или системы в целом. Автоматизация позволяет повысить производительность и качество системы, оптимизировать процессы управления, снизить затраты.

Так, система самостоятельно сможет регулировать количество загружаемых в нее книг, определять, есть ли среди новых книг повторные экземпляры уже имеющихся, и, если есть, добавлять их к общему количеству, и выдавать пользователю доступ к выбранным им книгам.

1.1 Особенности предметной области

Электронная библиотека – это относительно новая технология предоставления людям доступа к книгам без необходимости покидать пределы своего жилья.

Электронные библиотеки (ЭБ) представляют собой формы сложных распределенных информационных систем, предоставляющих новые возможности работы с неоднородной информацией, и рассматриваются как основа создания глобального распределенного хранилища знания.

Поскольку без онлайн-сервисов сейчас уже не обойтись, так как они прочно вошли в нашу жизнь, то использование электронной библиотеки для предоставления пользователям доступа к электронной базе данных многочисленных книг позволяет читателям познавать новое не листая страницы бумажной книги, а в новом формате на собственном устройстве получать информацию. Что выводит технологии предоставления информации обычному пользователю на новый уровень своего развития.

Электронные библиотеки обладают следующими преимуществами:

1. Доставляют библиотеку к пользователю, т. е. информация поступает непосредственно на «рабочий стол» пользователя, будь то дома или на работе. Теперь библиотека доступна там, где есть компьютер, подключенный к сети;
2. Электронные документы удобнее, чем их бумажные аналоги, искать и анализировать, поскольку практически любое слово в тексте может быть поисковым выражением;
3. Отсутствуют проблемы, связанные с невозможностью получить книги, занятые другими читателями: при расположении книг в сети, их количество не имеет значения;
4. Предоставляют возможность получить доступ к самым труднодоступным текстам; – экономят значительные финансовые средства, поскольку содержание электронных библиотек обходится значительно дешевле, чем традиционных. Это связано с тем, что хранение информации в электронном

виде дешевле, чем в бумажном, причем стоимость электронного хранения информации ежегодно снижается. Стоимость же содержания традиционных библиотек во всем мире возрастает; – в них легче поддерживать актуальность информации;

5. Информация в электронной библиотеке доступна всегда, в том числе в ночное время, а также в нерабочие и праздничные дни (в частности, исследования, проведенные в Британском университете, показали, что примерно половина посещений ЭБ приходится на то время, когда традиционные библиотеки уже закрыты). Таким образом, доступ к информации, расположенной в ЭБ, возможен в течение 24 часов в сутки семь дней в неделю;
6. Материалы в электронной библиотеке не могут оказаться недоступными из-за выдачи не тому пользователю, отправки книги на реставрацию или в переплет и т.д.;
7. Исключаются кражи книг и других материалов – в электронных библиотеках доступны новые многообразные виды информации (видео- и аудиоматериалы, 3D-объекты, интернет-источники и др.);
8. Пользователь электронной библиотеки, помимо того, что он обслуживается в телекоммуникационном режиме, чаще обращается ко всему пространству Интернета для получения необходимых ему документов и данных;
9. ЭБ доставляют информацию тем людям, для которых недоступны обычные библиотеки.

Главная цель программного обеспечения электронной библиотеки - обеспечить легкую и понятную для пользователя, организацию и доступ к книгам, чем это возможно в живом формате организации библиотеки.

Библиотека представляет собой хранилище книг. Основные ее функции - выдавать читателям книги и принимать их. Остальные функции относятся к обслуживанию процесса выдачи и приема. Экземпляр каждой книги, хранящейся в библиотеке, должен иметь индивидуальный номер.

1.2 Проблемы, возникающие в данной предметной области и перспективы их решения с использованием программных средств

В случае если в работе с онлайн-библиотекой наблюдаются некоторые проблемы, это повод предусмотреть пути их решения при проектировании программного обеспечения. Перечислим некоторые из вероятных проблем:

1. Потеря данных(полное или частичное удаление базы данных);
2. Взлом базы данных и утечка в сеть книг с ограниченным доступом;
3. Отсутствие постоянного доступа к книгам из базы данных библиотеки;
4. Сложность формирования библиотечной базы данных книг;
5. Быстрое старение компьютерных технологий;
6. Недолговечность веб-сайтов и соответственно электронных ресурсов, на них расположенных (средний срок жизни интернет-ресурса — от четырёх месяцев до двух лет);
7. Недолговечность современных машинных носителей информации;
8. Все еще дорогостоящая инфраструктура;
9. Зависимость от электричества и Интернета;
10. Проблемы защиты авторского права;
11. Отсутствие единых стандартов на форматы электронных книг;
12. Возможность неточностей и искажений в тексте из-за ошибок, возникающих при сканировании и распознавании бумажных источников.

Список этих проблем можно продолжать, но, тем не менее, электронные библиотеки уже стали реальностью современной жизни, т. е. преимущества электронных библиотек превалируют над их недостатками.

Эти и другие проблемы можно решить, если понаблюдать за процессами организации существующих аналогов электронных библиотек, после чего сделать выводы и устранить описанные выше проблемы.

1.3 Анализ существующих аналогов

В настоящее время интернет начинает играть всю более важную роль в нашей жизни. Именно в интернете мы можем найти самые свежие последние новости, необходимую нам информацию, музыку, кино, игры и т.д.

Как на смену обычной книге приходит электронная книга, так и на смену обычной библиотеке приходит электронная библиотека, то есть библиотека, где и каталог, и сами книги имеют электронный вид. И все большую популярность в последнее время набирают Интернет-библиотеки, виртуальные библиотеки, доступ к которым можно осуществлять из любой точки мира.

Сегодня в Интернете действуют свыше 4500 библиотек. Особой популярностью в последнее время также пользуется правовая и техническая электронная документация в виде информационно-правовых систем и систем по нормативной документации. Удобство поиска и использования необходимой информации в таких системах оценили уже многие. Во все мире активно идёт перевод фондов обычных библиотек в электронный вид. При этом электронную форму принимают не только книги, но и инфраструктура библиотеки.

Сегодня имеется три вида электронных библиотек:

1. Интернет-представительства реальных крупных библиотек страны
2. Платные ресурсы

В таких библиотеках чаще всего можно найти современные бестселлеры и популярные книги последних лет. Касательно оплаты здесь имеется два варианта: либо оплата происходит за доступ к каждой отдельно взятой книге(по сути это получается обычный книжный интернет-магазином с неплохими скидками), либо оплата берется за абонемент на месяц, полгода или год.

3. Полностью бесплатные интернет-хранилища.

Tululu.org – большая онлайн-библиотека, позволяющая не только читать книги в онлайн формате, но и скачивать их на свое устройство в TXT или JAR-форматах. Также в режиме чтения есть возможность запоминать страницу, на

которой Вы остановились и настраивать размер шрифта, цвет текста и фон, что очень удобно при чтении с мобильного телефона. Зарегистрированным пользователям доступна система комментирования и оценки прочитанных книг, а также возможность общаться на форуме. Что касается жанров произведений, то имеются как разделы популярной и классической литературы, так и более специфические, вроде каталогов научно-образовательных книг, компьютерной литературы, документальной и даже старинных книг.

Lib.ru - Русскоязычная электронная библиотека в Интернете, одна из первых и наиболее известных подобных библиотек в Рунете. Недостаток – дизайн сайта. Все реализовано на базе элементарного WEB 1.0 и пестрит десятками перекрестных ссылок. Практически все материалы на данном сайте доступны только в формате TXT, но поскольку данный формат поддерживается сегодня большинством устройств, то это не вызывает каки-либо проблем.

CoolLib.net

Проект работает по принципу Википедии, и каждый участник, зарегистрированный более двух дней назад, может принимать активное участие в формировании и упорядочивании библиотеки.

Особенности:

1. функция чтения онлайн для всех книг с настройкой шрифта и фона;
2. скачивание всех книг в формате EB2;
3. возможность чтения нескольких книг (функция "Книжная полка");
4. возможность самостоятельного добавления литературы;
5. наличие мобильной версии библиотеки.

1.4 Цель и задачи курсового проектирования

Цель работы заключается в разработке эффективной и удобной в использовании программы, позволяющей выбирать книги в библиотеке, добавлять

произведения в базу данных библиотеки. Для разработки программного обеспечения были поставлены следующие задачи:

1. Соблюдение основополагающих принципов ООП при разработке программы.
2. Реализация интуитивно понятного пользователю интерфейса.
3. Разделение возможностей читателя и библиотекаря.
4. Проверка корректности данных книги при добавлении.
5. Защищённость входа в программу (Для получения доступа к инструментам добавления книги и просмотра базы данных и передачи базы данных книг в интернет-магазин, пользователь должен ввести пароль).
6. Возможность отслеживать оставшееся количество дней до последнего дня хранения книг.
7. Реализация сортировки книг по жанру.

2 Моделирование и разработка системы на основе принципов ООП

2.1 Постановка задачи

В начале проектирования программы необходимо определить, как будет реализована одна из важнейших составляющих программы – диалог с пользователем. Было принято решение реализовать диалог программы с пользователем посредством интерфейса, реализованного при помощи системной библиотеки Windows Forms.

Согласно заданию необходимо реализовать программу работы онлайн-библиотеки.

Основные задачи, которые необходимо решить в рамках проектирования:

1. Создание удобного и понятного для пользователя интерфейса
2. Добавление книг в базу данных
3. Отслеживание времени окончания доступа к данной книге
4. Вывод списка произведений в виде таблицы
5. Возможность сортировать таблицу
6. Получение книги

В качестве дополнительной задачи:

7. Создание инструментария для библиотекаря, который будет содержать сведения и функции, доступные только ему.

Важной задачей остаётся создание удобного и интуитивно понятного интерфейса, с которым легко взаимодействовать любому пользователю.

Основное меню программы было решено реализовать в виде списка книг, где для работы необходимо выбрать жанр нажатием на кнопку соответствующего действия. Данное решение в программе реализуется при помощи библиотеки Windows Forms в языке C++ CLI. Пример вида основного меню программы (от лица пользователя) представлен на рисунке 1:

Рисунок 1 – Вид главного меню программы

Из рисунка 1 видно, что программа предполагает наличие двух пользователей, библиотекаря и читателя. Выбрав кнопку “Войти как читатель” открывается новая форма, в которой реализованы следующие функции, представленные на рисунке 2.

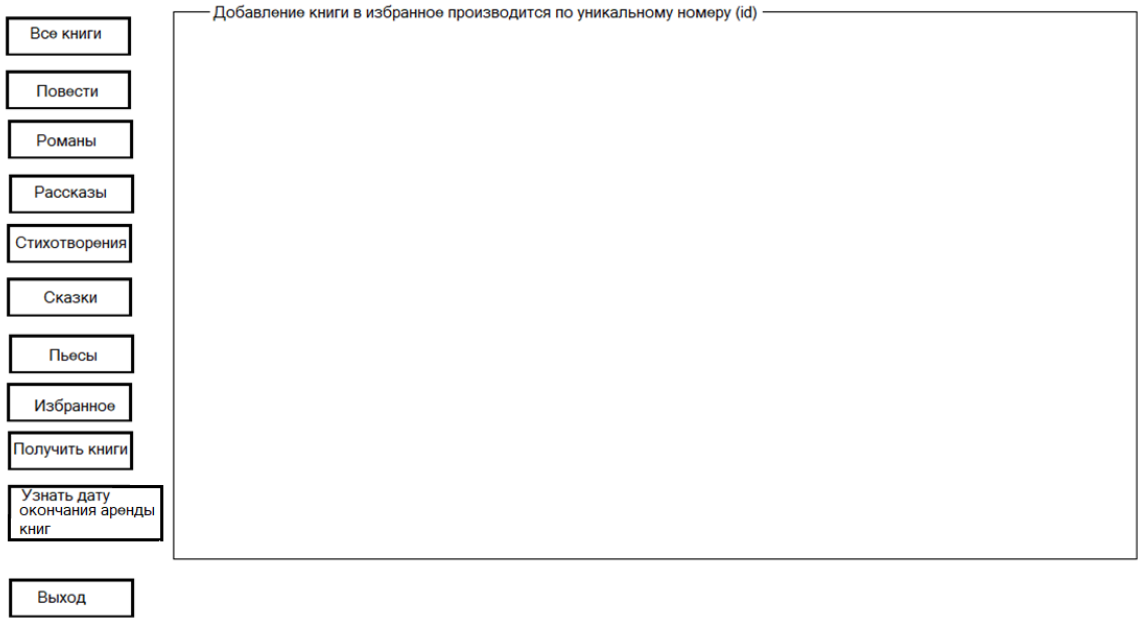


Рисунок 2 – Вид меню читателя

Из рисунка 2 видно, что программа предполагает выполнение восьми основных задач, а именно: отображение в виде таблицы всех произведений, их индивидуальный номер относительно списка, жанр, название, год издания автор, количество страниц и количество экземпляров.

Также программа предлагает выбрать жанр отображаемых произведений. Так, можно отображать: повести, романы, стихотворения, рассказы, сказки и пьесы. Кнопка “Избранное” реализует функцию, которая выводит в виде таблицы произведения, добавленные читателем.

Также пользователь может выбрать “войти как библиотекарь” и ввести пароль. Если пароль будет верным, то это вызовет форму библиотекаря, пример которой представлен на рисунке 3.

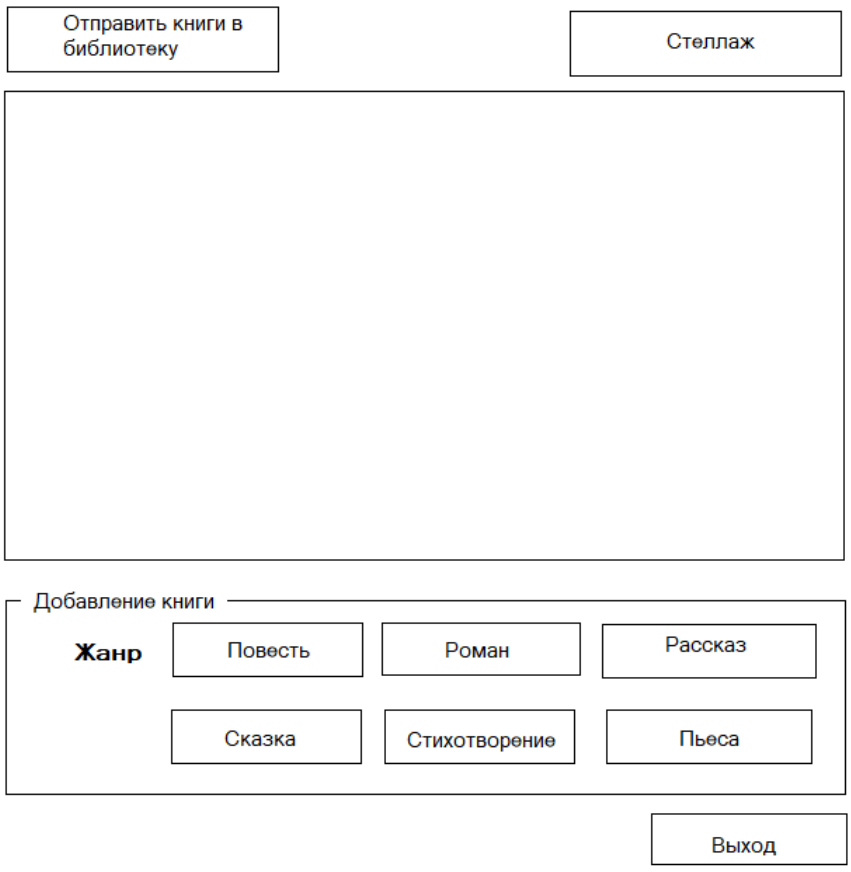


Рисунок 3 – Вид меню библиотекаря

Из рисунка 3 видно, что программа предполагает отображение списка книг в виде таблицы, передачу базы данных в библиотеку, добавление книги выбранного

жанра в базу данных. Пример внешнего вида окна, служащего для добавления книг в базу данных, представлен на рисунке 4.

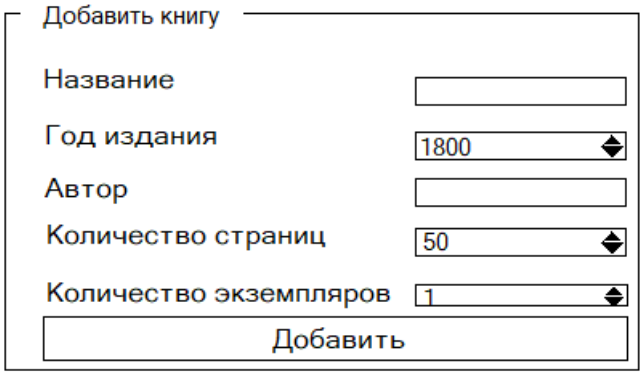


Рисунок 4 – Окно добавления книги

На рисунке 4 можно увидеть, что количество элементов, характерных для каждой книги, одинаково и соответствует количеству заголовков в таблице в меню читателя. Так как это окно идентично для каждого жанра, но собственное для каждого из них, то заголовок «Жанр» здесь не указывается. Пользователь(в данном случае, библиотекарь) выбирает жанр, щелкнув по соответствующей кнопке в меню библиотекаря.

2.2 Объектно-ориентированные принципы и компоненты

Объектно-ориентированное программирование (ООП)– это подход к построению программ, где главной отправной точкой служит объект с его свойствами и поведением.

Объекты создаются на основании класса, определённого в коде, и объединяют данные и действия, которые можно выполнять с данными, в одно целое.

Объектно-ориентированное программирование обладает следующими свойствами:

1) инкапсуляция;

2) наследование.

3) полиморфизм.

1. Инкапсуляция — это соединение в одной структуре данных (классе) данных и операций над данными в сочетании со скрытием ненужной для использования этих данных информации.

Инкапсуляция повышает степень уровня абстракции программы, то есть данные класса и реализация его функций лежат ниже уровня абстракции и для написания программы информация о них не требуется. Инкапсуляция также позволяет изменить реализацию класса без модификации основной части программы, если интерфейс остался прежним. Инкапсуляция позволяет использовать класс в другом окружении и быть уверенным, что он не испортит не принадлежащие ему области памяти. А также создавать библиотеки классов для применения во многих программах.

2. Наследование — свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствованной функциональностью.
3. Полиморфизм — свойство системы, которое позволяет использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

В процессе разработки архитектуры программного решения необходимо разделить реализацию проекта на отдельные малосвязанные между собой составляющие.

Диаграмма классов, реализуемых в проекте, представлена на рисунке 5.

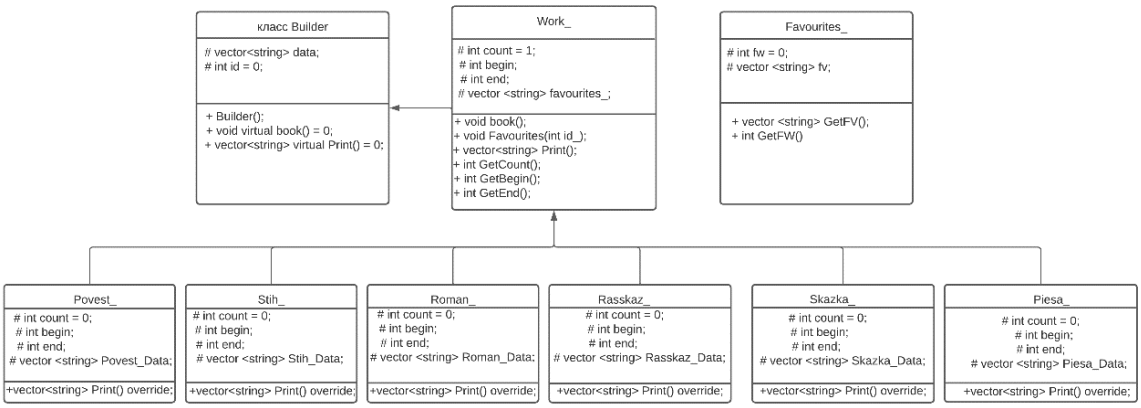


Рисунок 5 – Диаграмма классов, необходимая для пользовательского интерфейса

2.3 Разработка классов

Основная информация о реализованных в ходе написания программы классах, методах и функциях, а также об особенностях препроцессора представлена соответственно в таблицах 1, 2, 3.

Таблица 1 – Список классов

Класс	Примечания
Class Librarian	Отвечает за интерфейс библиотекаря
Class Builder	Абстрактный класс
Class Favourites_	Отвечает за считывание информации о книгах, выбранных читателем
class Work_: public Builder	Наследуется от абстрактного класса Builder и реализует проверку базы данных на корректность
class Povest_: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом povest из базы данных.
class Stih_: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом stih из базы данных.
class Rasskaz_: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом rasskaz из базы данных.

Продолжение Таблицы 1

Класс	Примечания
class Roman_: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом roman из базы данных.
class Skazka_: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом skazka из базы данных.
Class Piesa: public Work_	Наследуется от класса Work_ и реализует методы, ориентированные на работу с типом piesa из базы данных.
Class Temp	Отвечает за открытие потока и создания GUID’a

Таблица 2 – Список свойств и методов классов

Класс	Метод	Примечания
Librarian	vector<string> Print_Lib_data();	Возвращает вектор строчного типа, считанный из базы данных
Librarian	string GetPassword();	Возвращает строчный тип пароля библиотекаря
Librarian	int GetCount()	Возвращает количество элементов в коллекции librarian
Builder	Void virtual print() = 0;	Виртуальный метод класса
Favourites_	vector <string> GetFV();	Возвращает вектор строчного типа
Favourites_	int GetFW()	Возвращает количество элементов
Work_: public Builder_	Void Favourites(int id_);	Добавляет элемент из коллекции в “избранные”
Work_: public Builder_	Vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных
Work_: public Builder_	Int GetCount();	Возвращает размер коллекции
Work_: public Builder_	Int GetBegin();	Возвращает первый элемент коллекции

Продолжение Таблицы 2

Класс	Метод	Примечания
Work_: public Builder_	Int GetEnd();	Возвращает индекс последнего элемента коллекции
Povest_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа povest
Stih_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа stih
Roman_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа roman
Rasskaz_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа rasskaz
Skazka_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа skazka
Piesa_ : public Work_	vector<string> Print() override;	Возвращает вектор строчного типа, считанный из базы данных типа piesa
Temp	Guid D(void);	Метод возвращает целочисленное значение, отвечающее за время доставки

Таблица 3 – Список объявляемых функций

Имя и тип	Примечания
void TransferData();	Передача библиотекарской базы данных в читательскую базу данных.
void ReturnData();	Восстановление базы данных, в случае ошибки или потери данных.
vector<Heads> DivideRow(vector<string> s, int count);	Заполнение вектора типа структуры Heads, деление строки из базы данных на элементы.

Список макроопределений препроцессора

#define FILE_FAVOURITES_NAME "Favourites.txt" - Имя файла, содержащего информацию о книгах, добавленных читателем в Избранное
#define BOOKS_FILE "Data.txt" - Имя файла, содержащего информацию о доступных читателю книгах
#define MAIN_BOOKS "DefaultData.txt" - Имя файла, содержащего информацию обо всех книгах, которая доступна библиотекарю
#define SEARCH_EXP_NEW R"(\w{2,17}\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Общее регулярное выражение
#define SEARCH_POVEST R"((Povest)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Повести
#define SEARCH_ROMAN R"((Roman)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Романы
#define SEARCH_RASSKAZ R"((Rasskaz)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Рассказы
#define SEARCH_STIH R"((Stihotvorenie)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Стихотворения
#define SEARCH_PIESA R"((Piesa)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Пьесы
#define SEARCH_SKAZKA R"((Skazka)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" - Регулярное выражение для группы Сказки
#define BITSTRING R"((\w{2,30}\s)|(\w{3,15})|(\d{2,4})|(\d{1}))" - Регулярное выражение для разбиения строки на подстроки

В процессе написания программы создается несколько файлов. В файлах содержится информация о книгах онлайн-библиотеках читателя и библиотекаря.

Для удобства хранения была сформирована структура, в которой будут содержаться информация о книгах онлайн-библиотеки, среди которых жанр книги, название, год издания, автор, количество страниц и экземпляров.

```
struct Heads
{
	string Janr = "";
	string Nazvanie = "";
	string Year = "";
	string Author = "";
	string Pages = "";
	string Ekzemps = "";
	};
```

Выше было сказано, что функционал программы разделен для двух пользователей. Так первая часть программы отвечает за читателя. Для него были реализованы ряд функций и классов.

Главным в программе является абстрактный класс Builder. Конструктор класса builder считывает данные из текстового файла и заполняет ими вектор строчного типа data. Свойства класса имеют модификатор доступа protected: к ним относится вектор строчного типа data и целочисленный тип id.

От абстрактного класса с модификатором доступа public наследуется класс Work_. Класс имеет переопределяемый метод Print() и Favourites(), который принимает аргумент целочисленного типа. Также класс имеет геттеры для получения первого индекса, последнего и количества. Свойства класса имеют модификатор доступа protected: вектор строчного типа favourites_ и переменные целочисленного типа count, begin, end.

Классы Povest_, Roman_, Rasskaz_, Stih_, Skazka, Piesa_ наследуются с модификатором доступа public от класса Work_. Они имеют переопределенную функцию Print(), которая возвращает вектор строчного типа. Так же класс наследует свойства и методы от родительского класса. Все классы отличаются конструктором по умолчанию. Конструктор выбирает, благодаря регулярному выражению, прописанного для каждого класса, нужные ему данные.

Класс Favourites_, реализован для возвращения вектора строчного типа, базы данных «Избранных» книг читателя.

Вторая часть программы реализована для работы библиотекаря. Для него был разработан класс Librarian. Конструктор класса Librarian() – считывает данные из текстового файла и заполняет ими вектор строчного типа librarian_data. Методы класса: Print_Lib_data() – возвращает вектор строчного типа, GetPassword и GetCount – являются геттером для возвращения пароля библиотекаря и количества элементов в коллекции.

Две части программы формируют полноценную систему работы онлайн-библиотеки, включающей обзор от лица двух пользователей – читателя и библиотекаря с разными возможностями взаимодействия с интерфейсом

программы. В программе осуществляется обновление базы данных, сортировка книг по желанию читателя. Библиотекарь может добавлять книги, проводить проверку хранилища библиотеки, обновлять базу данных онлайн-библиотеки для читателя с обычными правами доступа.

3. Реализация системы на общесистемном языке программирования

3.1. Выбор средств программной реализации

В число современных сред разработки программного обеспечения, подходящей для решения поставленной задачи, можно отнести Microsoft Visual Studio 2022. Для разработки данного программного обеспечения нужно выбрать такой язык программирования, который использует принципы объектно-ориентированного программирования, в нашем случае было принято решение использовать язык C++.

Microsoft Visual Studio 2022 - интегрированная среда разработки, включающая инструментальные средства для проектирования, кодирования, транслирования, отладки и выполнения программ. Microsoft Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня.

Диалог программы с пользователем будет осуществляться при помощи графического пользовательского интерфейса (GUI). В качестве инструмента для создания клиентского оконного приложения будет использоваться набор управляемых библиотек платформы Windows Forms, входящая в состав фреймфорка .NET. Платформа обеспечивает один из самых эффективных способов создания классических приложений с помощью визуального конструктора в Visual Studio IDE, упрощая создание при помощи размещения визуальных элементов управления путём перетаскивания на необходимую форму (представление окна приложения).

Для того чтобы в процессе разработки была возможность работать с технологией Windows Form и использовать C++ в качестве основного языка программирования, необходимо использовать C++ CLI — модификацию ЯП от Microsoft, входящую в состав языков, совместимых с платформой разработки .NET, с возможностью запуска в среде CLR.

3.2 Алгоритм выполнения программы

Главным классом в выполнении программы является общедоступный тип «LoginForm», который определяет статический публичный метод «Run», который запускает процесс создания графического интерфейса и т. д.

В момент запуска функции «main» вызывается метод «Run», происходит отрисовка первой формы «LoginForm», после чего пользователь, может обратится к форме «AutorizationForm» и «LibraryForm». Каждая последующая форма использует базовый класс «Builder_». При вызове «AutorizationForm» подгружается класс Librarian, отвечающий за блок графического представления авторизации. При вызове «Librarian» подгружается базовый класс и дочерние от него классы «Book», который отвечает за сортировку и корректное отображение в графическом блоке «LibraryForm».

«LibrarianForm1», «LibrarianForm2», «LibrarianForm3», «LibrarianForm4», «LibrarianForm5», «LibrarianForm6» - отвечают за создание нового объекта, который добавляется в общую коллекцию.

В зависимости от выбранного пользователя профиля устанавливается нужное представление в качестве основного окна.

3.3 Модульная структура программы

Модульное программирование представляет собой принцип организации программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определённым правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок.

Программный код часто разбивается на несколько файлов, каждый из которых компилируется отдельно от остальных. Такая модульность программного кода позволяет значительно уменьшить время перекомпиляции при изменениях,

вносимых лишь в небольшое количество исходных файлов, и упрощает групповую разработку.

Система взаимодействия пользователь – книга, состоит их двух самостоятельных элементов: форма библиотекаря и форма читателя.

Функционал реализуемой программы рассредоточен по формам, каждая из которых предоставляет некоторый набор инструментов, для взаимодействия с программой.

В зависимости от выбранного аккаунта, устанавливается нужное представление в качестве основного окна. Загруженные сервисы, подгружаются в качестве приватных полей внутри всех классов, наследуемых от базового «Windows:Form». Менеджер служб используется при создании экземпляра формы, в момент вызова конструктора, он предоставляет провайдера на запрашиваемые зависимости.

На рисунке 6 представлена модульная структура программы, где каждый модуль является отдельной формой и представлен разными блоками.

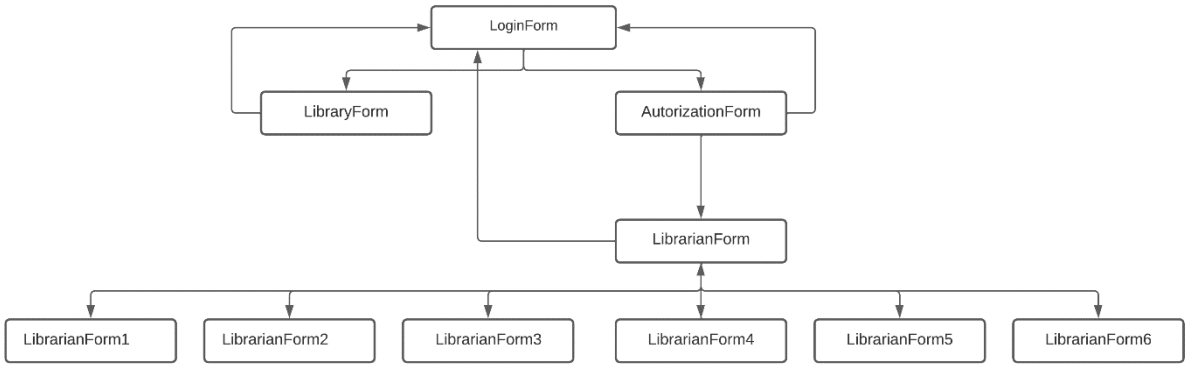


Рисунок 6 – Модульная структура программы

На рисунке 6 можно увидеть, что LibraryForm и AutorizationForm – производные формы из LoginForm. Читатель взаимодействует с формой LibraryForm, доступ к которой свободен. Из этой формы он может вернуться, нажав специальную кнопку, обратно в форму LoginForm. Если пользователь захочет зайти как библиотекарь(попасть в форму библиотекаря – LibrarianForm), то для доступа к

данной форме необходимо будет, попав в форму AutorizationForm, пройти авторизацию, введя пароль, и, если он будет введен верно, авторизация пройдет успешно, и откроется форма LibrarianForm. Из формы LibrarianForm пользователь, как и из формы AutorizationForm сможет вернуться в форму LoginForm, нажав на специальную кнопку. Через форму LibrarianForm пользователь может открывать ее производные формы - LibrarianForm1, LibrarianForm2, LibrarianForm3, LibrarianForm4, LibrarianForm5, LibrarianForm6.

3.4 Описание диалога с пользователем

При запуске программы пользователя встречает меню, предоставляющее возможность входа в систему под одним из двух пользователей.

На рисунке 7 представлено окно входа в программу, реализуемое с помощью формы LoginForm.

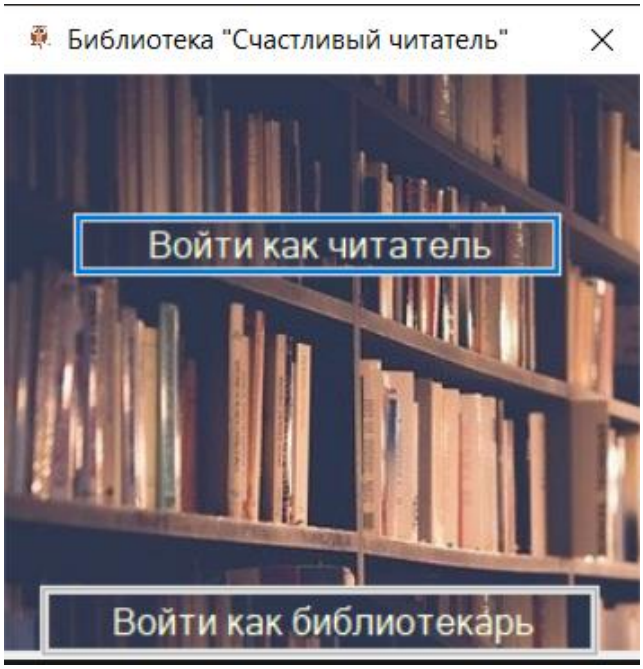


Рисунок 7 – Меню входа в систему

При помощи клика мышки на выбранную кнопку осуществляется переход на следующую форму, которая будет зависеть от выбора самого пользователя.

Так, если пользователь нажмёт на кнопку “Войти как читатель”, то он будет направлен на главное меню читателя.

На рисунке 7 представлен вид главного меню читателя.

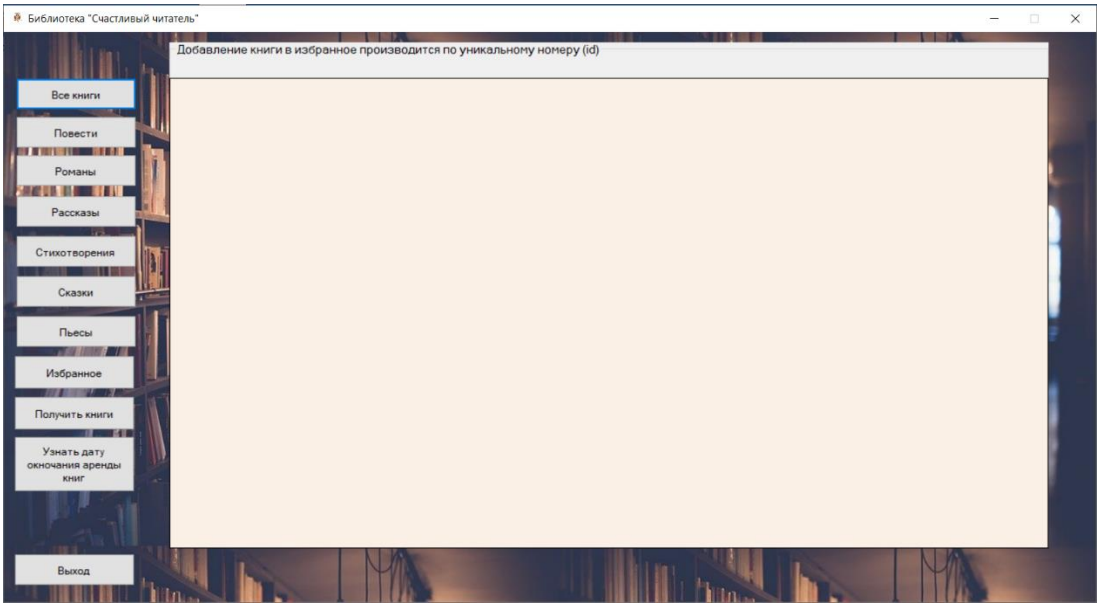


Рисунок 7 – Главное меню читателя

После открытия меню, у пользователя, который теперь уже является читателем, есть ряд возможностей, выполнение которых осуществляется кликом по определенной кнопке.

Так, “Все книги” выведет информацию обо всех хранящихся книгах в виде таблицы.

На рисунке 8 представлен вид таблицы, если читатель выберет «Все книги».

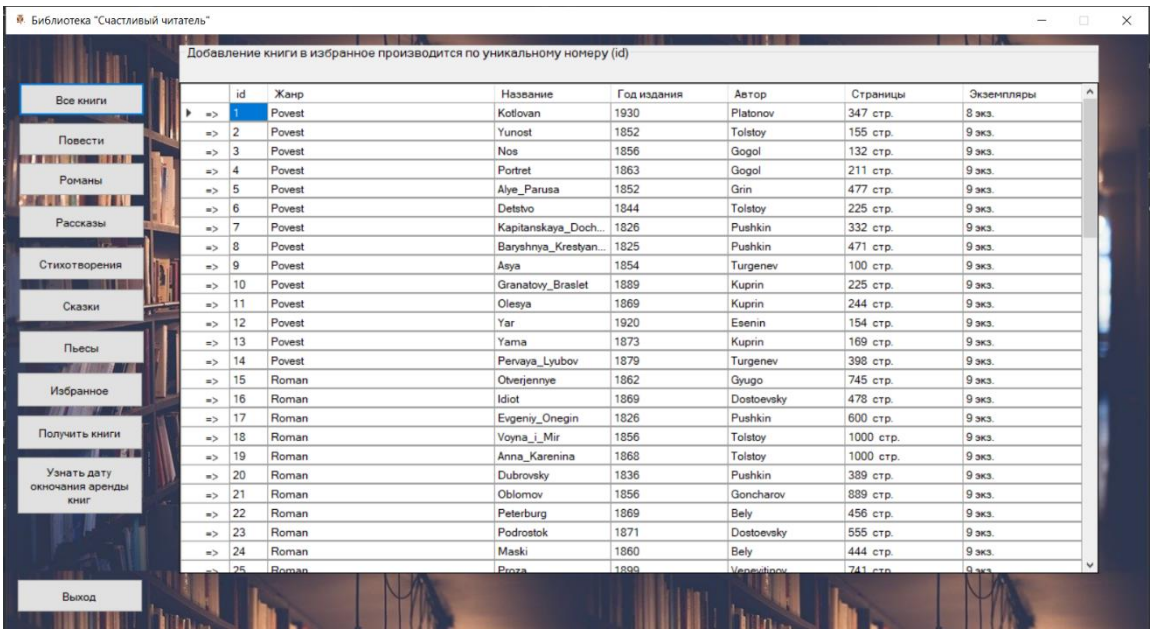


Рисунок 8 – Вывод результата клика по кнопке «Все книги»

Читатель так же может выбрать отображение какой-либо одной категории. Для этого необходимо кликнуть на соответствующие кнопки (повести, рассказы, романы, стихотворения, сказки, пьесы).

На рисунке 9 представлен вид таблицы, если читатель выберет жанр «Повести».

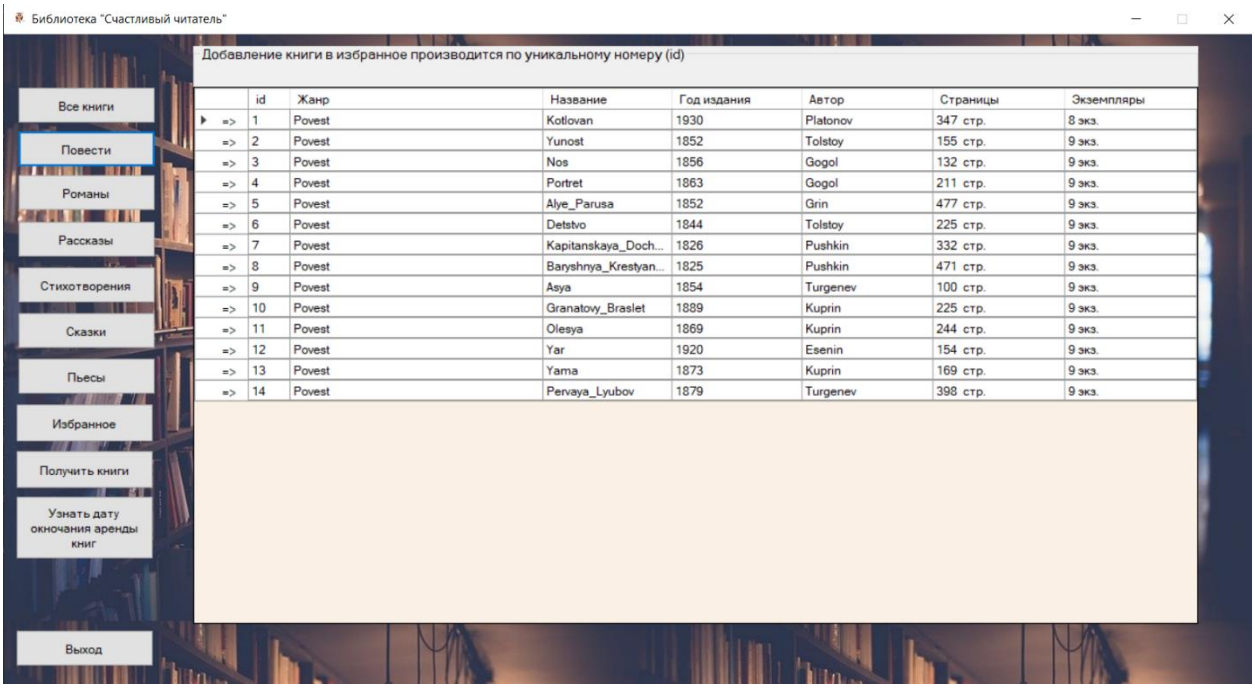
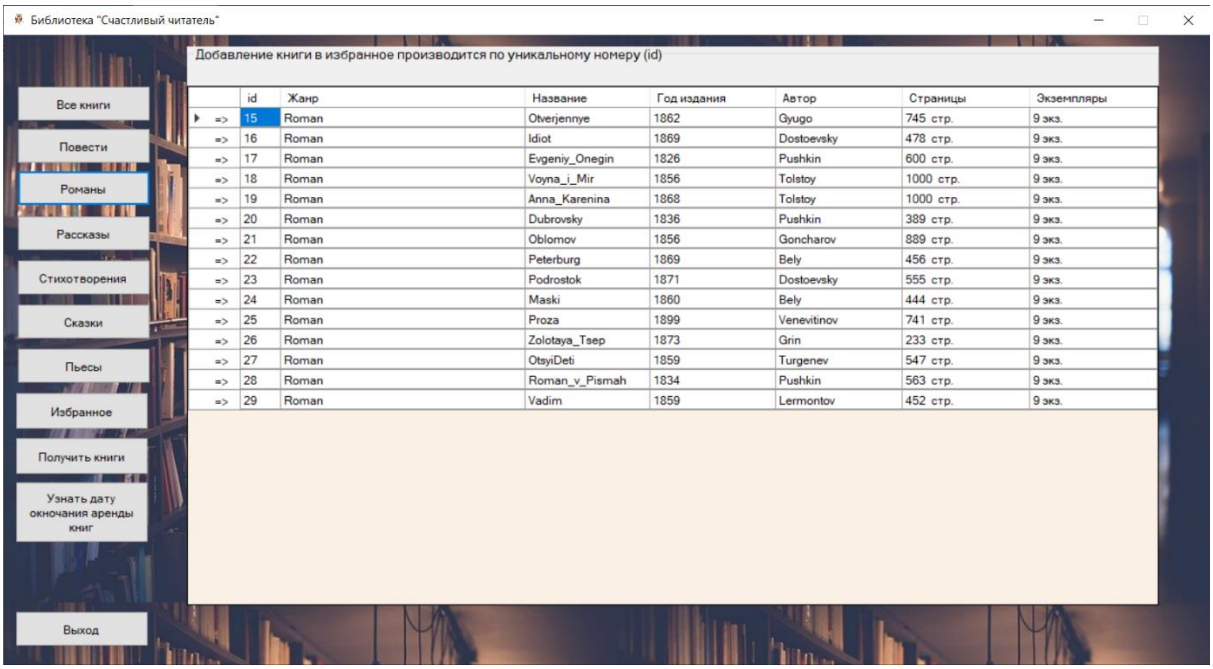


Рисунок 9 - Вывод результата клика по кнопке «Повести»

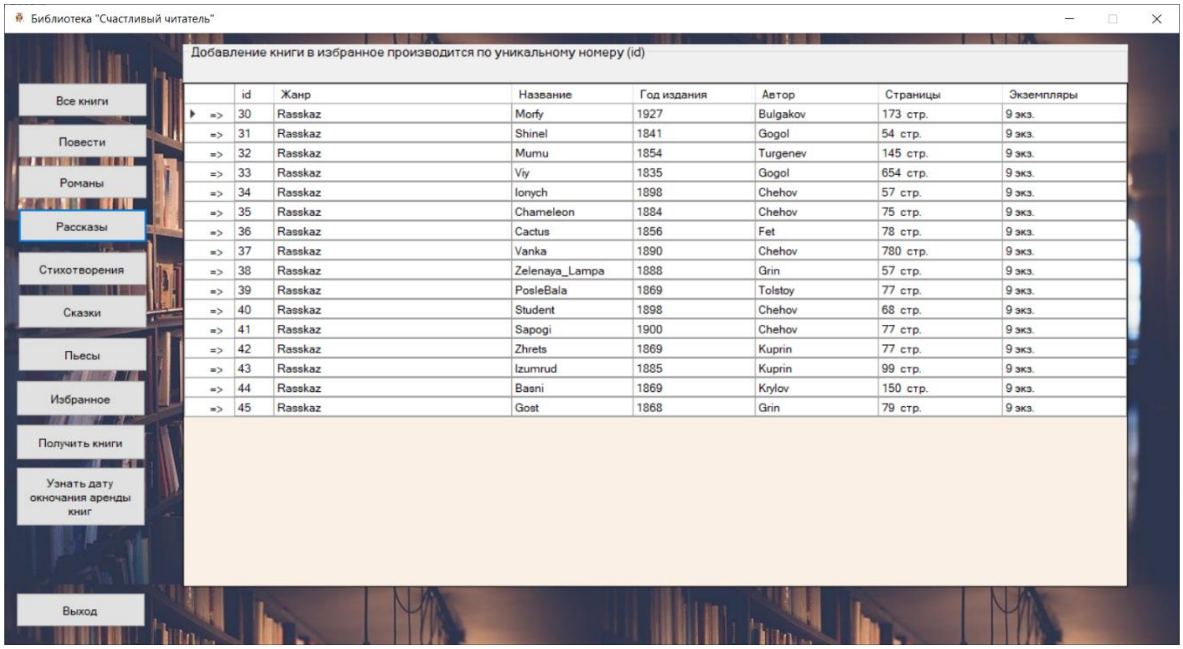
На рисунке 10 представлен вид таблицы, если читатель выберет жанр «Романы».



	id	Жанр	Название	Год издания	Автор	Страницы	Экземпляры
=>	15	Roman	Otverjennye	1862	Gyugo	745 стр.	9 экз.
=>	16	Roman	Idiot	1869	Dostoevsky	478 стр.	9 экз.
=>	17	Roman	Evgeniy_Onegin	1826	Pushkin	600 стр.	9 экз.
=>	18	Roman	Voyna_i_Mir	1856	Tolstoy	1000 стр.	9 экз.
=>	19	Roman	Anna_Karenina	1868	Tolstoy	1000 стр.	9 экз.
=>	20	Roman	Dubrovsky	1836	Pushkin	389 стр.	9 экз.
=>	21	Roman	Oblomov	1856	Goncharov	889 стр.	9 экз.
=>	22	Roman	Peterburg	1869	Bely	456 стр.	9 экз.
=>	23	Roman	Podrostok	1871	Dostoevsky	555 стр.	9 экз.
=>	24	Roman	Maski	1860	Bely	444 стр.	9 экз.
=>	25	Roman	Proza	1899	Venevitinov	741 стр.	9 экз.
=>	26	Roman	Zolotaya_Tsep	1873	Grin	233 стр.	9 экз.
=>	27	Roman	OtsyiDeti	1859	Turgenev	547 стр.	9 экз.
=>	28	Roman	Roman_v_Pismah	1834	Pushkin	563 стр.	9 экз.
=>	29	Roman	Vadim	1859	Lermontov	452 стр.	9 экз.

Рисунок 10 - Вывод результата клика по кнопке «Романы»

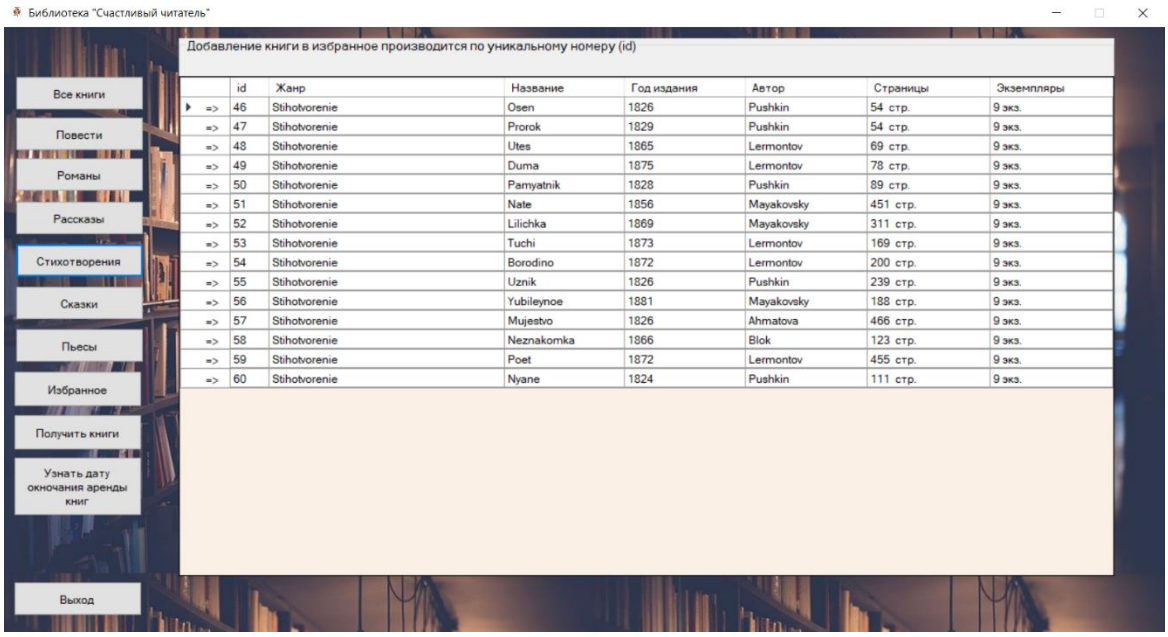
На рисунке 11 представлен вид таблицы, если читатель выберет жанр «Рассказы».



	id	Жанр	Название	Год издания	Автор	Страницы	Экземпляры
=>	30	Rasskaz	Morfy	1927	Bulgakov	173 стр.	9 экз.
=>	31	Rasskaz	Shinel	1841	Gogol	54 стр.	9 экз.
=>	32	Rasskaz	Mumu	1854	Turgenev	145 стр.	9 экз.
=>	33	Rasskaz	Viy	1835	Gogol	654 стр.	9 экз.
=>	34	Rasskaz	Ionych	1898	Chehov	57 стр.	9 экз.
=>	35	Rasskaz	Chameleon	1884	Chehov	75 стр.	9 экз.
=>	36	Rasskaz	Cactus	1856	Fet	78 стр.	9 экз.
=>	37	Rasskaz	Vanka	1890	Chehov	780 стр.	9 экз.
=>	38	Rasskaz	Zelenaya_Lampa	1888	Grin	57 стр.	9 экз.
=>	39	Rasskaz	PosleBala	1869	Tolstoy	77 стр.	9 экз.
=>	40	Rasskaz	Student	1898	Chehov	68 стр.	9 экз.
=>	41	Rasskaz	Sapogi	1900	Chehov	77 стр.	9 экз.
=>	42	Rasskaz	Zhrets	1869	Kuprin	77 стр.	9 экз.
=>	43	Rasskaz	Izumrud	1885	Kuprin	99 стр.	9 экз.
=>	44	Rasskaz	Basni	1869	Krylov	150 стр.	9 экз.
=>	45	Rasskaz	Gost	1868	Grin	79 стр.	9 экз.

Рисунок 11 - Вывод результата клика по кнопке «Рассказы»

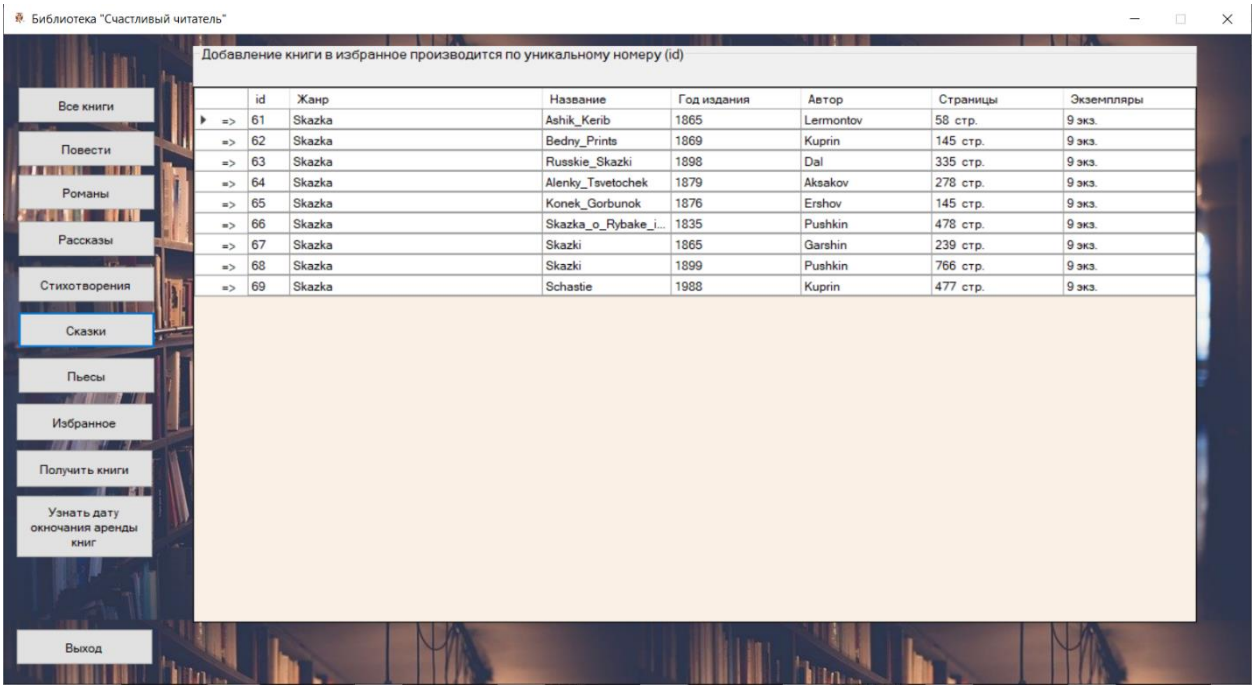
На рисунке 12 представлен вид таблицы, если читатель выберет жанр «Стихотворения».



	id	Жанр	Название	Год издания	Автор	Страницы	Экземпляры
=>	46	Stihotvorenie	Osen	1826	Pushkin	54 стр.	9 экз.
=>	47	Stihotvorenie	Prorok	1829	Pushkin	54 стр.	9 экз.
=>	48	Stihotvorenie	Utes	1865	Lermontov	69 стр.	9 экз.
=>	49	Stihotvorenie	Duma	1875	Lermontov	78 стр.	9 экз.
=>	50	Stihotvorenie	Pamyatnik	1828	Pushkin	89 стр.	9 экз.
=>	51	Stihotvorenie	Nate	1856	Mayakovsky	451 стр.	9 экз.
=>	52	Stihotvorenie	Lilichka	1869	Mayakovsky	311 стр.	9 экз.
=>	53	Stihotvorenie	Tuchi	1873	Lermontov	169 стр.	9 экз.
=>	54	Stihotvorenie	Borodino	1872	Lermontov	200 стр.	9 экз.
=>	55	Stihotvorenie	Uznik	1826	Pushkin	239 стр.	9 экз.
=>	56	Stihotvorenie	Yubileynoe	1881	Mayakovsky	188 стр.	9 экз.
=>	57	Stihotvorenie	Mujestvo	1826	Ahmatova	466 стр.	9 экз.
=>	58	Stihotvorenie	Neznakomka	1866	Blok	123 стр.	9 экз.
=>	59	Stihotvorenie	Poet	1872	Lermontov	455 стр.	9 экз.
=>	60	Stihotvorenie	Nyane	1824	Pushkin	111 стр.	9 экз.

Рисунок 12 - Вывод результата клика по кнопке «Стихотворения»

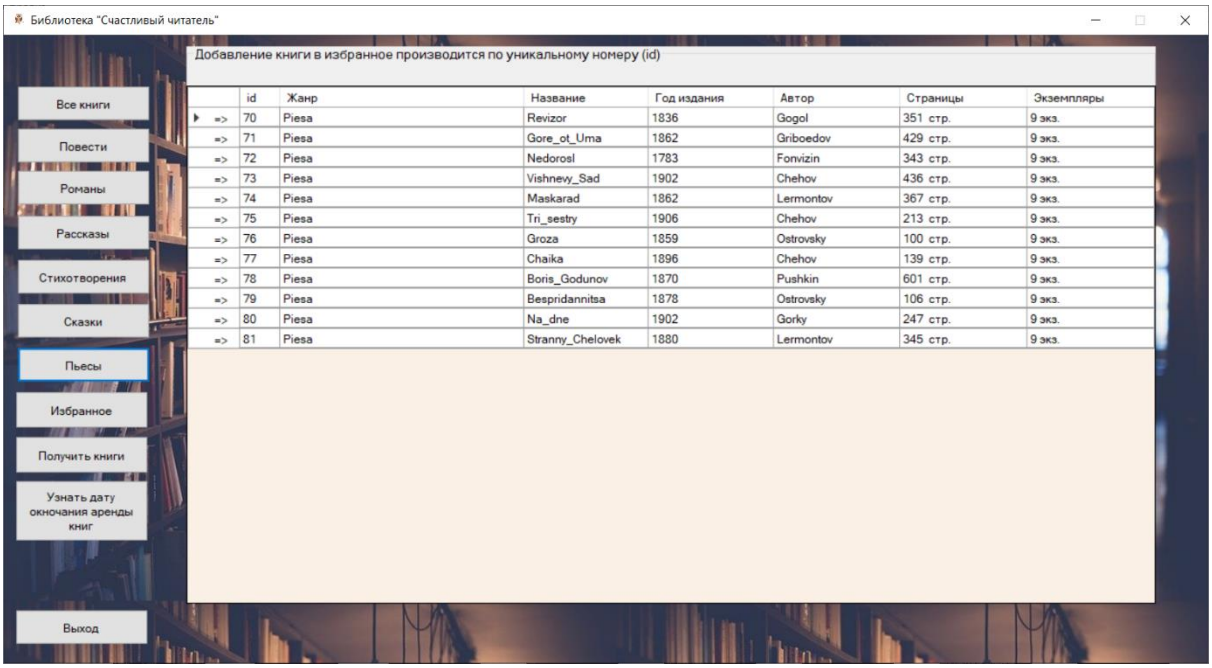
На рисунке 13 представлен вид таблицы, если читатель выберет жанр «Сказки».



	id	Жанр	Название	Год издания	Автор	Страницы	Экземпляры
=>	61	Skazka	Ashik_Kerib	1865	Lermontov	58 стр.	9 экз.
=>	62	Skazka	Bedny_Prints	1869	Kuprin	145 стр.	9 экз.
=>	63	Skazka	Russkie_Skazki	1898	Dal	335 стр.	9 экз.
=>	64	Skazka	Alenky_Tsvetochek	1879	Aksakov	278 стр.	9 экз.
=>	65	Skazka	Konek_Gorbunok	1876	Ershov	145 стр.	9 экз.
=>	66	Skazka	Skazka_o_Rybake_i...	1835	Pushkin	478 стр.	9 экз.
=>	67	Skazka	Skazki	1865	Garshin	239 стр.	9 экз.
=>	68	Skazka	Skazki	1899	Pushkin	766 стр.	9 экз.
=>	69	Skazka	Schastie	1988	Kuprin	477 стр.	9 экз.

Рисунок 13 - Вывод результата клика по кнопке «Сказки»

На рисунке 14 представлен вид таблицы, если читатель выберет жанр «Пьесы».



	id	Жанр	Название	Год издания	Автор	Страницы	Экземпляры
=>	70	Piesa	Revizor	1836	Gogol	351 стр.	9 экз.
=>	71	Piesa	Gore_ot_Uma	1862	Griboedov	429 стр.	9 экз.
=>	72	Piesa	Nedorosl	1783	Fonvizin	343 стр.	9 экз.
=>	73	Piesa	Vishnevy_Sad	1902	Chehov	436 стр.	9 экз.
=>	74	Piesa	Maskarad	1862	Lermontov	367 стр.	9 экз.
=>	75	Piesa	Tri_sestry	1906	Chehov	213 стр.	9 экз.
=>	76	Piesa	Groza	1859	Ostrovsky	100 стр.	9 экз.
=>	77	Piesa	Chaika	1896	Chehov	139 стр.	9 экз.
=>	78	Piesa	Boris_Godunov	1870	Pushkin	601 стр.	9 экз.
=>	79	Piesa	Bespridannitsa	1878	Ostrovsky	106 стр.	9 экз.
=>	80	Piesa	Na_dne	1902	Gorky	247 стр.	9 экз.
=>	81	Piesa	Stranny_Chelovek	1880	Lermontov	345 стр.	9 экз.

Рисунок 14 - Вывод результата клика по кнопке «Пьесы»

Если читатель захочет получить какую-либо книгу, он может добавить ее в Избранное, дважды кликнув по ячейке из столбцов id и ###. После этого количество экземпляров книг уменьшается, или если книга была в одном экземпляре, то она перестаёт отображаться в таблице.

Используя рисунок 15, на примере книги с индексом 64, представленной на данный момент в единственном экземпляре, можно убедиться в корректности работы алгоритма добавления книги в Избранное.

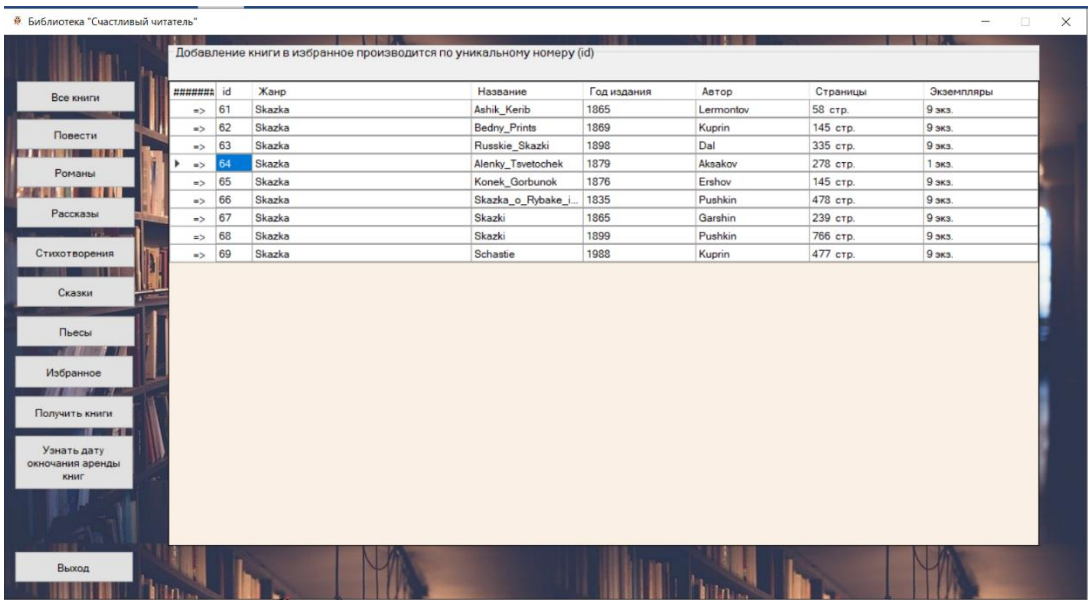


Рисунок 15 – Добавление читателем книги в Избранное

Нажав на кнопку Избранное, читатель сможет увидеть те книги, которые он добавил в данный раздел, что представлено на рисунке 16.

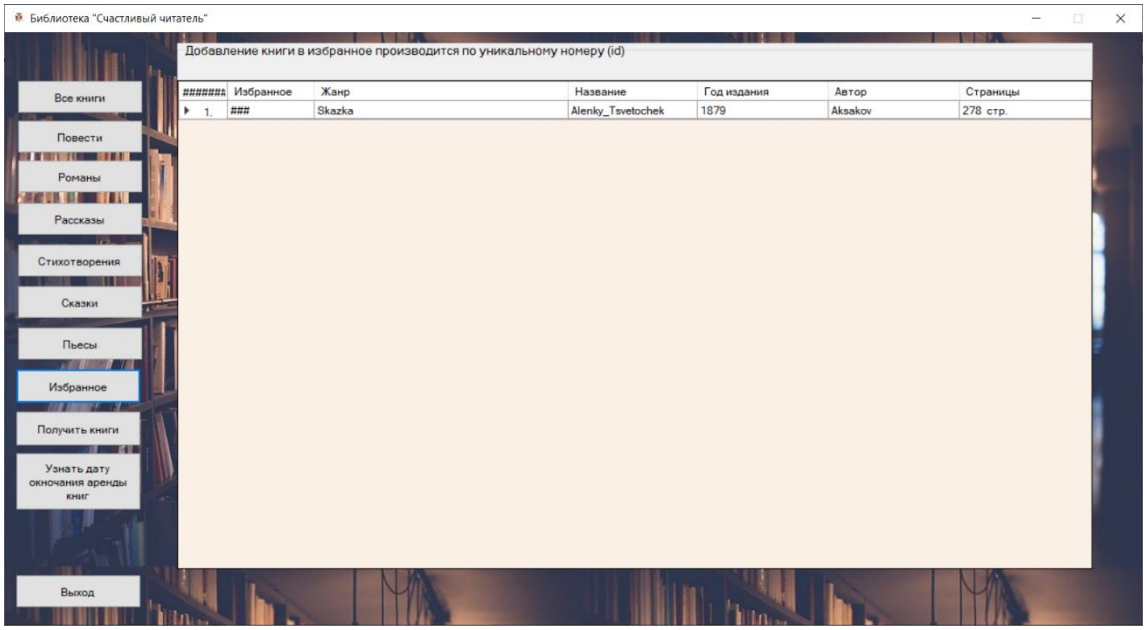


Рисунок 16 - Таблица добавленных книг читателем

После добавления первой книги в Избранное, читатель может нажать на кнопку “Получить книги”, что очищает базу добавленного и отображает время до окончания доступа к полученным книгам относительно компьютерного времени.

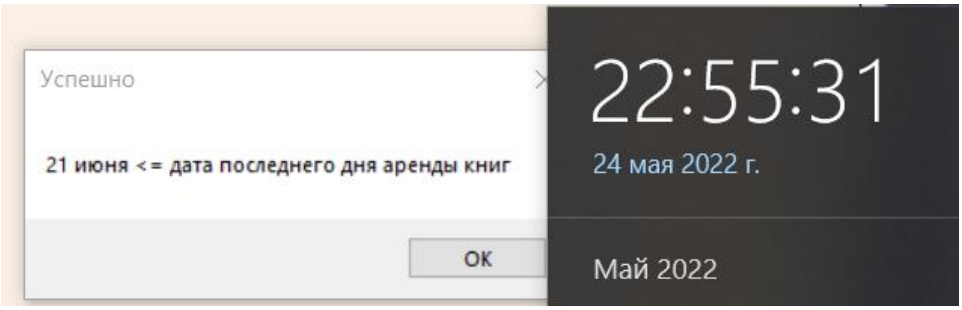


Рисунок 17 – Дата, до наступления которой читатель может сохранять доступ к книге

После закрытия окна, можно также проверять статус заказа. Так, нажав на кнопку “Узнать дату окончания аренды книг” будет вызвано окно, в котором будет отображено количество оставшихся дней до приостановления доступа.

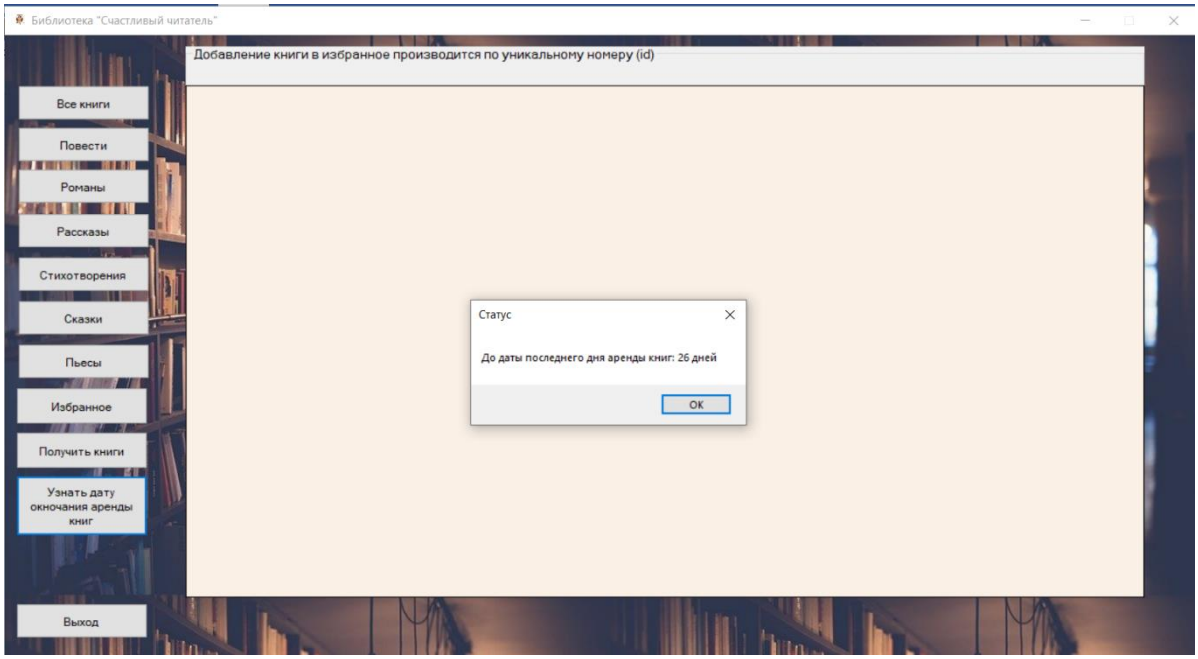


Рисунок 18 – Проверка статуса аренды книги

После нажатия на кнопку «Получить книги» читатель получит уведомление о том, что книги добавлены в его профиль, то есть он получил доступ к ним.

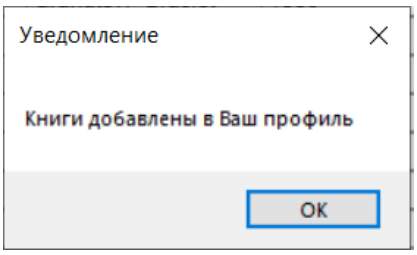


Рисунок 19 – Уведомление о разрешении на доступ к книгам

После наступления даты окончания доступа, читатель получит следующие уведомления, представленные на рисунке 20.

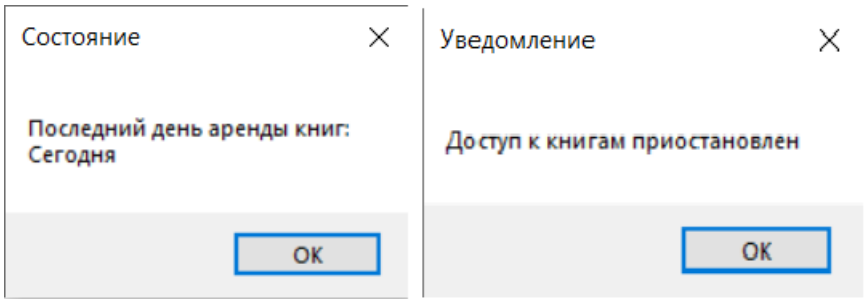


Рисунок 20 – Сообщения, получаемые читателем

Также в случае действия читателя не по выше описанному алгоритму, а по иному поведению, к примеру, при ошибочном нажатии на ячейку таблицы, или попытке узнать дату окончания аренды книг, имея пустой раздел Избранного, для каждого случая будут открыты диалоговые окна с указанием на ошибку.

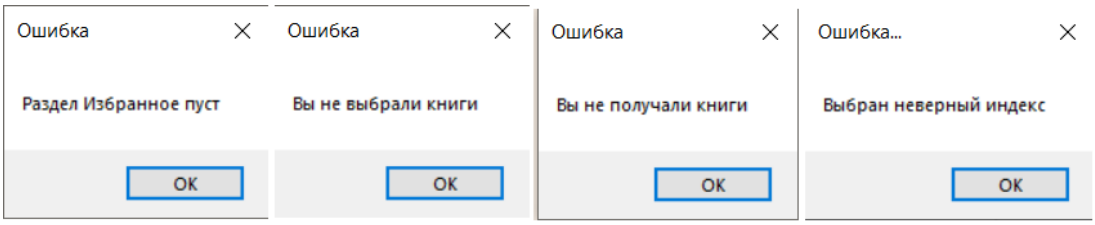


Рисунок 21 – Список ошибок в читательской форме

Кнопка в левом нижнем углу «Выход» отвечает за выход из режима читателя и возвращает к выбору пользователя.

После нажатия на кнопку “Войти как библиотекарь” открывается окно с авторизацией библиотекаря.

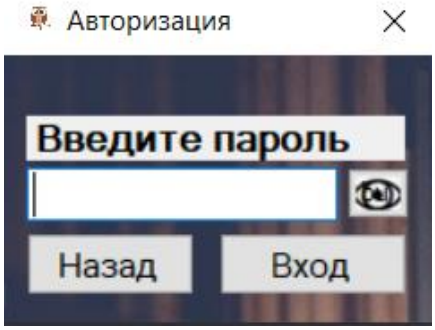


Рисунок 22 – Авторизация библиотекаря

Если пароль введен неверно, открывается диалоговое окно с информацией об этом.

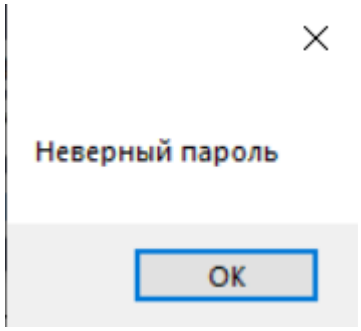


Рисунок 23 – Сообщение о неверно введенном пароле

Кнопка с картинкой “Глаз” позволяет при клике на нее раскрыть символы пароля.

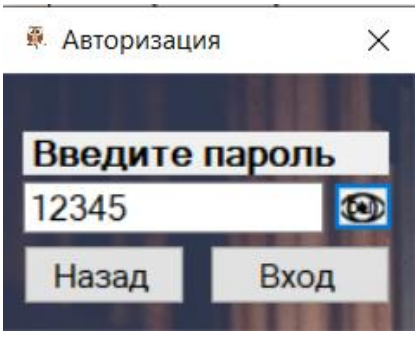


Рисунок 24 – Раскрытие пароля

В случае успешного ввода пароля и нажатия кнопки “Вход” пользователю программы откроется окно библиотекаря.

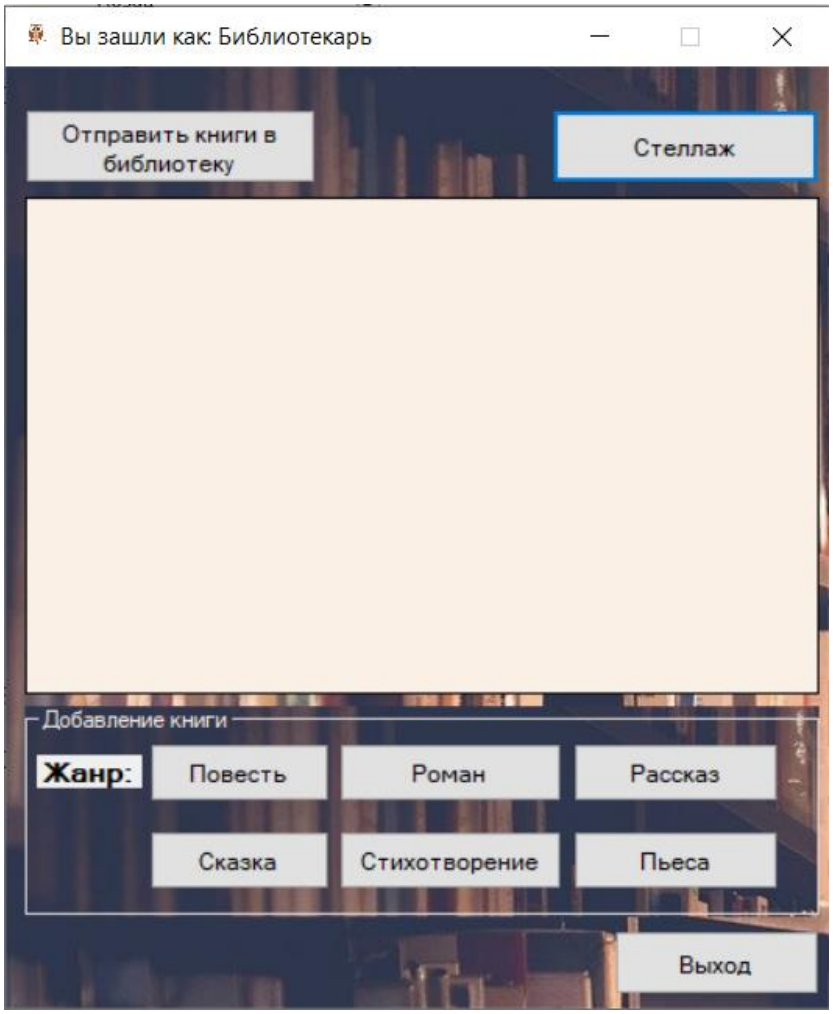


Рисунок 25 – Окно библиотекаря

Для отображения книг на стеллаже библиотекарю необходимо нажать на кнопку “Стеллаж”, после чего произойдет отрисовка таблицы книг.

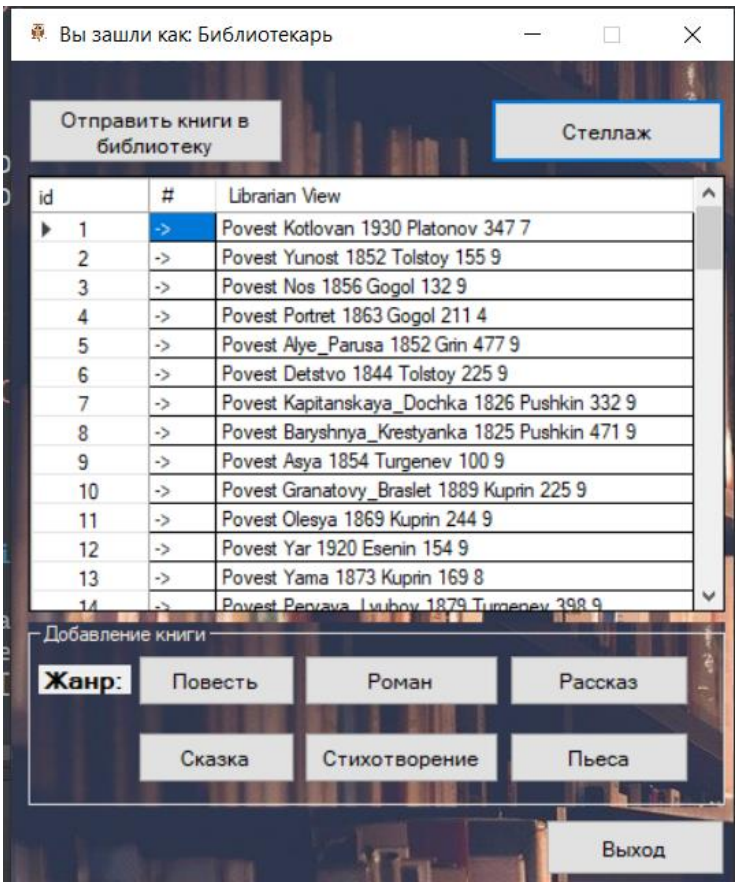


Рисунок 26 – Отображение книг на стеллаже

Библиотекарь может добавить новое произведение путём нажатия на одну из шести кнопок: “Повесть”, “Роман”, “Рассказ”, ”Сказка”, ”Стихотворение”, “Пьеса”. При нажатии на любую кнопку из них открывается диалоговое окно, в котором можно создать книгу жанра, соответствующего информации на кнопке. После чего книга будет отображаться в таблице.

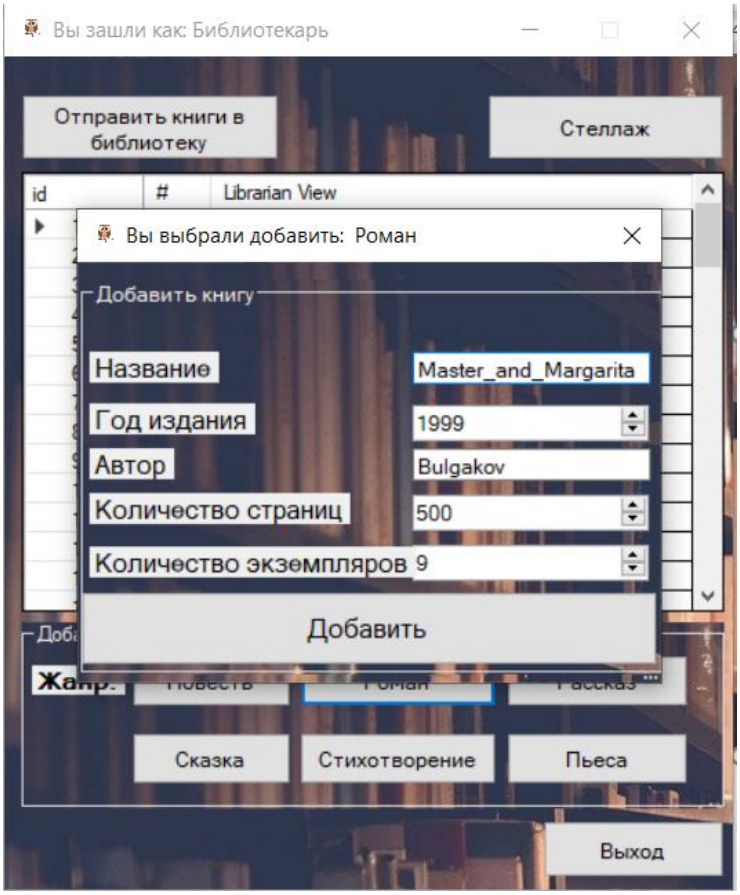


Рисунок 27 – Добавление новой книги в библиотеку

После выполнения данной операции открывается диалоговое окно, оповещающее об успешности добавления книги.

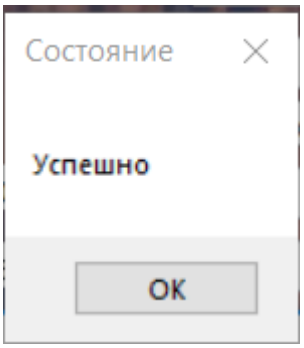


Рисунок 28 – Оповещение о состоянии

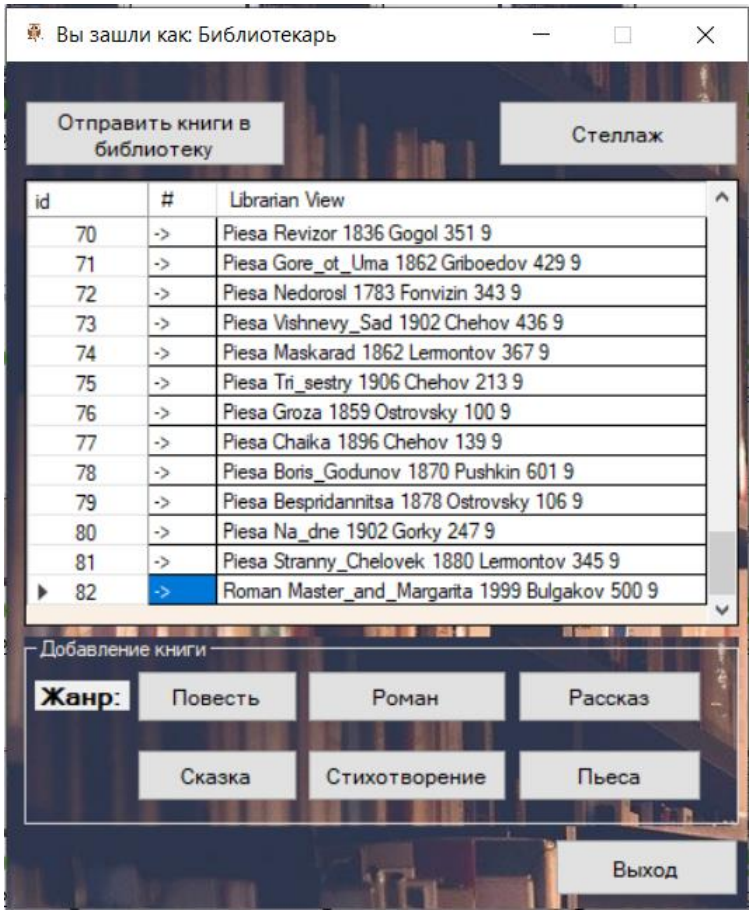


Рисунок 29 – Отображение новой книги в библиотеке

Последней операции библиотекаря является передача данных о хранилище книг библиотеке, благодаря чему читатель сможет увидеть созданную книгу и по желанию приобрести к ней доступ.

=>	75	Piesa	Tri_sestry	1906	Chehov	213 стр.	9 экз.
=>	76	Piesa	Groza	1859	Ostrovsky	100 стр.	9 экз.
=>	77	Piesa	Chaika	1896	Chehov	139 стр.	9 экз.
=>	78	Piesa	Boris_Godunov	1870	Pushkin	601 стр.	9 экз.
=>	79	Piesa	Bespridannitsa	1878	Ostrovsky	106 стр.	9 экз.
=>	80	Piesa	Na_dne	1902	Gorky	247 стр.	9 экз.
=>	81	Piesa	Stranny_Chelovek	1880	Lermontov	345 стр.	9 экз.
=>	82	Roman	Master_and_Marga...	1999	Bulgakov	500 стр.	9 экз.

Рисунок 30 – Результат передачи книги в библиотеку

Также как в форме читателя, в форме библиотекаря предусмотрена кнопка выхода, что позволит закрыть данную форму и вернуться в меню выбора пользователей.

Заключение

В ходе курсового проектирования на языке C++ с использованием разработки Visual Studio 2022 была разработана программа, реализующая работу электронной библиотеки.

Разработанная программа предоставляет пользователю систему с удобным интерфейсом с разными вариантами входа и зависящего от этого внешнего вида рабочего окна, для выполнения операций, выбранных пользователем. Организован контроль ошибок при их возникновении.

Все требования к программе выполнены успешно. Реализованы следующие аспекты: иерархия классов, соответствующая и использующая три основных принципа ООП, необходимые для корректной работы функции. Сортировка вектора по заданным условиям. Разработана структура, предоставляющая удобное обращение к элементам, добавление новых элементов, открытие потоков. Было создано два файла с произведениями. Использована среда CLR для создания форм пользовательского интерфейса.

Список использованной литературы

1. Документация по языку Си++ - https://docs.microsoft.com/ru-ru/cpp

2. Норенков И.П. Основы автоматизированного проектирования: Учеб. для вузов. – 3-е изд., перераб. и доп. / И.П. Норенков. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2006. – 448с.

3. Лафоре Роберт Объектно-Ориентированное программирование в С++: Учеб для вузов. – 4-е издание.

4. Курипта О.В. Основы программирования и алгоритмизации: практикум / О.В.Курипта, О.В. Минакова, Д.К. Проскурин; Воронежский ГАСУ. – Воронеж, 2015. – 132 с.

5. Надейкина Л.А. Программирование: учебное пособие. – М.: МГТУ ГА, 2017. – 84 с.

6. Документация по среде CLR - https://docs.microsoft.com/ru-ru/dotnet/standard/clr

Приложение

Листинг программы на C++

Classes.h

```
#pragma once
#include <iostream>
#include <regex>
#include <vector>
#include <thread>
#include <algorithm>
#include <fstream>
#include <string>
#include <chrono>
#include <ctime>

using namespace std;

#define FILE_FAVOURITES_NAME "Favourites.txt"
#define BOOKS_FILE "Data.txt"                        // Имена Файлов
#define MAIN_BOOKS "DefaultData.txt"

#define SEARCH_EXP_NEW R"(\w{2,17}\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //Общее
регулярное выражение
#define SEARCH_POVEST R"((Povest)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Повести
#define SEARCH_ROMAN R"((Roman)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Романы
#define SEARCH_RASSKAZ R"((Rasskaz)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Рассказы
#define SEARCH_STIH R"((Stihotvorenie)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Стихотворения
#define SEARCH_PIESA R"((Piesa)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Пьесы
#define SEARCH_SKAZKA R"((Skazka)\s\w{2,30}\s\d{4}\s\w{3,15}\s\d{2,4}\s\d{1})" //
Регулярное выражение для группы Сказки
#define BITSTRING R"((\w{2,30}\s)|(\w{3,15})|(\d{2,4})|(\d{1}))" //Регулярное выражения
для разбиения строки на подстроки

enum Head_name
{
	Janr = 1,
	Nazvanie = 2,
	Year = 3,
	Author = 4,
	Pages = 5,
	Ekzemps = 6,
};

struct Heads
{
	string Janr = "";
	string Nazvanie = "";
	string Year = "";
	string Author = "";
	string Pages = "";
	string Ekzemps = "";
};

class Librarian
```

```
{
public:
	Librarian();
	vector<string> Print_Lib_data();
	string GetPassword() { return Password; };
	int GetCount() { return count; }
private:
	const string Password = "12345";
	vector <string> librarian_data;
	int count = 0;
};
class Builder
{
public:
	void virtual book() = 0;
	std::vector<std::string> virtual Print() = 0;
	Builder();
protected:
	vector<string> data;
	int id = 0;
};

class Favourites_
{
public:
	Favourites_();
	vector <string> GetFV();
	int GetFW() { return fw; }
private:
	int fw = 0;
	vector <string> fv;
};
class Work_ : public Builder
{
public:
	void book() override;
	void Favourites(int id_);
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> favourites_;
	int count = 1;
	int begin;
	int end;
};
class Povest_ : public Work_
{
public:
	Povest_();
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Povest_Data;
	int count = 0;
	int begin = 0;
	int end = 0;
};
class Stih_ : public Work_
{
public:
	Stih_();
```

```
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Stih_Data;
	int count = 0;
};
class Roman_ : public Work_
{
public:
	Roman_();
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Roman_Data;
	int count = 0;
};
class Rasskaz_ : public Work_
{
public:
	Rasskaz_();
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Rasskaz_Data;
	int count = 0;
};
class Skazka_ : public Work_
{
public:
	Skazka_();
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Skazka_Data;
	int count = 0;
};
class Piesa_ : public Work_
{
public:
	Piesa_();
	std::vector<std::string> Print() override;
	int GetCount() { return count; }
	int GetBegin() { return begin; }
	int GetEnd() { return end; }
protected:
	vector <string> Piesa_Data;
	int count = 0;
};
std::vector<Heads> DivideRow(std::vector<std::string> s, int count);
```

Classes.cpp

```
#include "Classes.h"
#include "Functions.h"
```

```
using namespace System;
using namespace System::Threading::Tasks;
Librarian::Librarian()
{
	ifstream File(BOOKS_FILE);
	if (File.is_open())
	{
		count = 0;
		while (!File.eof()) {
			string temp;
			getline(File, temp);
			this->librarian_data.push_back(temp);
			count++;
		}
	}
	else { System::Windows::Forms::MessageBox::Show("Файл повреждён", "Ошибка"); }
	File.close();
}


std::vector<std::string> Librarian::Print_Lib_data()
{
	return librarian_data;
}

Favourites_::Favourites_()
{
	ifstream File(FILE_FAVOURITES_NAME);
	if (!File.is_open()) throw exception("File read error");
	while (!File.eof()) {
		fw++; string temp; //
		getline(File, temp);
		this->fv.push_back(temp);
	}
	File.close();
}

vector <string> Favourites_::GetFV()
{
	return fv;
}

Builder::Builder()
{
	ifstream File(BOOKS_FILE);
	if (!File.is_open()) throw exception("File read error");
	while (!File.eof()) {
		id++; string temp; //
		getline(File, temp);
		this->data.push_back(temp);
	}
	File.close();
}

void Work_::book()
{
	regex regular(SEARCH_EXP_NEW);
	smatch find_word;
	vector<string>::iterator it = data.begin();
	int i = 0;
	count = id;
	while (i < id) {
		if (!regex_match(data[i], find_word, regular)) {
			data.erase(data.begin() + i);
			id--; i--;
```

```
			}
			i++;
		}
}

void Work_::Favourites(int id_)
{
	fstream File(BOOKS_FILE); fstream File_Favourites(FILE_FAVOURITES_NAME,
ios_base::app); // открытие файлов
	id_--;
	Favourites_ favourites_; auto valueFavourites = favourites_.GetFV();
	string temp; string buff;
	temp = (data[id_].c_str()[data[id_].size() - 1]);
	buff = data[id_];
	int count = stoi(temp);
	if (count == 1) {
		buff.replace(buff.size() - 1, buff.size(), "1");
		if (valueFavourites[NULL] == "") {
			File_Favourites << buff;
		}
		else
		{
			File_Favourites << endl << buff;
		}
		data.erase(data.begin() + id_);
	}
	else {
		buff.replace(buff.size() - 1, buff.size(), "1");
		if (valueFavourites[NULL] == "") {
			File_Favourites << buff;
		}
		else
		{
			File_Favourites << endl << buff;
		}
		count--; string temp2 = to_string(count);
		this->data[id_].replace(data[id_].size() - 1, data[id_].size(), temp2);
	}
	File.close(); File_Favourites.close();
	ofstream File_New(BOOKS_FILE, ios_base::trunc);
	if (!File_New.is_open()) throw exception("File read error");
	for (int i = 0; i < data.size(); i++) {
		if (i == data.size() - 1) File_New << data[i];
		else File_New << data[i] << endl;
	}
	File_New.close();
}

std::vector<string> Work_::Print()
{
	return data;
}

Povest_::Povest_()
{
	bool temp = true;
	smatch find_world;
	regex regular(SEARCH_POVEST);
	for (int i = 0; i < id; i++)
	{
		if (regex_search(data[i], find_world, regular)) {
			if (temp) begin = i;
			count++;
			Povest_Data.push_back(data[i]);
			end = i;
```

```
                temp = false;
            }

        }
}

std::vector<std::string> Povest_::Print()
{
        return Povest_Data;
}

Stih_::Stih_()
{
        bool temp = true;
        smatch find_world;
        regex regular(SEARCH_STIH);
        for (int i = 0; i < id; i++)
        {
            if (regex_search(data[i], find_world, regular)) {
                if (temp) begin = i;
                count++;
                Stih_Data.push_back(data[i]);
                end = i;
                temp = false;
            }
        }
}

std::vector<std::string> Stih_::Print()
{
        return Stih_Data;
}

Roman_::Roman_()
{
        bool temp = true;
        smatch find_world;
        regex regular(SEARCH_ROMAN);
        for (int i = 0; i < id; i++)
        {
            if (regex_search(data[i], find_world, regular)) {
                if (temp) begin = i;
                count++;
                Roman_Data.push_back(data[i]);
                end = i;
                temp = false;
            }
        }
}

std::vector<std::string> Roman_::Print()
{
        return Roman_Data;
}

Rasskaz_::Rasskaz_()
{
        bool temp = true;
        smatch find_world;
        regex regular(SEARCH_RASSKAZ);
        for (int i = 0; i < id; i++)
        {
            if (regex_search(data[i], find_world, regular)) {
                if (temp) begin = i;
                count++;
```

```
                    Rasskaz_Data.push_back(data[i]);
                    end = i;
                    temp = false;
                }
        }
}

std::vector<std::string> Rasskaz_::Print()
{
        return Rasskaz_Data;
}
Skazka_::Skazka_()
{
        bool temp = true;
        smatch find_world;
        regex regular(SEARCH_SKAZKA);
        for (int i = 0; i < id; i++)
        {
                if (regex_search(data[i], find_world, regular)) {
                    if (temp) begin = i;
                    count++;
                    Skazka_Data.push_back(data[i]);
                    end = i;
                    temp = false;
                }
        }
}

std::vector<std::string> Skazka_::Print()
{
        return Skazka_Data;
}
Piesa_::Piesa_()
{
        bool temp = true;
        smatch find_world;
        regex regular(SEARCH_PIESA);
        for (int i = 0; i < id; i++)
        {
                if (regex_search(data[i], find_world, regular)) {
                    if (temp) begin = i;
                    count++;
                    Piesa_Data.push_back(data[i]);
                    end = i;
                    temp = false;
                }
        }
}

std::vector<std::string> Piesa_::Print()
{
        return Piesa_Data;
                                                }
```

Functions.cpp

```
#include "Functions.h"
#include "Classes.h"
#include "LibraryForm.h"
using namespace System;

void TransferData()
```

```
{
	ifstream File_Default(MAIN_BOOKS);
	vector<string> temp_data; string temp;
	if (!File_Default.is_open()) throw exception("File read error");
	while (!File_Default.eof()) {
		getline(File_Default, temp);
		temp_data.push_back(temp);
	}
	File_Default.close();
	ofstream File_Data(BOOKS_FILE, ios_base::trunc);
	if (!File_Data.is_open()) throw exception("File read error");
	for (int i = 0; i < temp_data.size(); i++) {
		if (i == temp_data.size() - 1) File_Data << temp_data[i];
		else File_Data << temp_data[i] << endl;
	}

	File_Data.close();
}

void ReturnData()
{
	std::vector<std::string> ret;
	Work_ work;
	work.book();
	ret = work.Print();
	ofstream File_New(BOOKS_FILE, ios_base::trunc);
	if (!File_New.is_open()) throw exception("File read error");
	for (int i = 0; i < ret.size(); i++) {
		if (i == ret.size() - 1) File_New << ret[i];
		else File_New << ret[i] << endl;
	}
}

std::string Stos(System::String^ s)
{
	using namespace System::Runtime::InteropServices;
	const char* chars =
		(const char*)(Marshal::StringToHGlobalAnsi(s)).ToPointer();
	std::string os(chars);
	return os;
}

System::String^ Convert_string_To_String(std::string& os)
{
	System::String^ s;
	s = gcnew System::String(os.c_str());
	return s;
}

vector<Heads> DivideRow(vector<string> s, int count)
{
	Heads heads;
	vector <Heads> heads_v;
	regex reg(BITSTRING);
	smatch sm;
	for (int i = 0; i < count; i++)
	{
		heads_v.push_back(heads);
		int iter = 0;
		while (regex_search(s[i], sm, reg))
		{
			switch (iter)
			{
			case 0:
				heads_v[i].Janr = sm.str();
```

```
                        break;
                  case 1:
                        heads_v[i].Nazvanie = sm.str();
                        break;
                  case 2:
                        heads_v[i].Year = sm.str();
                        break;
                  case 3:
                        heads_v[i].Author = sm.str();
                        break;
                  case 4:
                        heads_v[i].Pages = sm.str();
                        break;
                  case 5:
                        heads_v[i].Ekzemps = sm.str();
                        break;
                  default:
                        break;
                  }
                  s[i] = sm.suffix();
                  iter++;
            }
      }
      return heads_v;
                                          }
```

Functions.h

```
#pragma once
#include <iostream>
#include "LibraryForm.h"
using namespace System;
void TransferData();
void ReturnData();
System::String^ Convert_string_To_String(std::string& v);
                        std::string Stos(System::String^ s);
```

LoginForm.h

```
#include "LoginForm.h"
#include "LibraryForm.h"
#include "AutorizationForm.h"
#include "Functions.h"
using namespace System;
using namespace System::Windows::Forms;

[STAThreadAttribute]
int main(array<String^>^ args)
{
      Application::EnableVisualStyles();
      Application::SetCompatibleTextRenderingDefault(false);
      CurseProject::LoginForm form;
      Application::Run(% form);
}
System::Void CurseProject::LoginForm::AdminButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
      Form::Hide();
      AutorizationForm^ form1 = gcnew AutorizationForm();
```

```
	form1->Show();
}

System::Void CurseProject::LoginForm::ChitatelButton_Click(System::Object^ sender,
System::EventArgs^ e)
{
	ReturnData();
	Form::Hide();
	LibraryForm^ form2 = gcnew LibraryForm();
	form2->Show();
}
```

LibraryForm.cpp

```
#include "LibraryForm.h"
#include "Classes.h"
#include "Functions.h"
#include "LoginForm.h"
using namespace System::Windows::Forms;
using namespace System::Threading;
using namespace System::Threading::Tasks;
using namespace System::Runtime::InteropServices;
System::Void	CurseProject::LibraryForm::ALL_Click(System::Object^	sender,
System::EventArgs^ e)
{
	dataGridData->Rows->Clear();
	dataGridData->Columns->Clear();
	Work_ work; auto v = work.Print(); work.book();
	dataGridData->RowCount = work.GetCount();
	if (v[NULL] == "") {
		MessageBox::Show("Библиотека закрыта, ведется уборка помещения", "Ошибка");
		dataGridData->Rows->Clear();
		dataGridData->Columns->Clear();
	}
	else
	{
		Headers();
		dataGridData->AutoResizeRows();
		Show();
	}
}

System::Void	CurseProject::LibraryForm::Povest_Click(System::Object^	sender,
System::EventArgs^ e)
{
	dataGridData->Rows->Clear();
```

```
	dataGridData->Columns->Clear();
	Povest_ povest; Work_ work;
	auto v = work.Print();
	dataGridData->RowCount = povest.GetCount();
	if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
	else
	{
		Headers();
		dataGridData->AutoResizeRows();
		ShowPovest();
	}
}
System::Void	CurseProject::LibraryForm::Stih_Click(System::Object^	sender,
System::EventArgs^ e)
{
	dataGridData->Rows->Clear();
	dataGridData->Columns->Clear();
	Stih_ stih; Work_ work; auto v = work.Print();
	dataGridData->RowCount = stih.GetCount();
	if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
	else
	{
		Headers();
		dataGridData->AutoResizeRows();
		ShowStih();
	}
}
System::Void	CurseProject::LibraryForm::Skazka_Click(System::Object^	sender,
System::EventArgs^ e)
{
	dataGridData->Rows->Clear();
	dataGridData->Columns->Clear();
	Skazka_ skazka; Work_ work; auto v = work.Print();
	dataGridData->RowCount = skazka.GetCount();
	if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
	else
	{
		Headers();
		dataGridData->AutoResizeRows();
		ShowSkazka();
	}
```

```
}
System::Void        CurseProject::LibraryForm::Piesa_Click(System::Object^        sender,
System::EventArgs^ e)
{
      dataGridData->Rows->Clear();
      dataGridData->Columns->Clear();
      Piesa_ piesa; Work_ work; auto v = work.Print();
      dataGridData->RowCount = piesa.GetCount();
      if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
      else
      {
            Headers();
            dataGridData->AutoResizeRows();
            ShowPiesa();
      }
}

System::Void        CurseProject::LibraryForm::Roman_Click(System::Object^        sender,
System::EventArgs^ e)
{
      dataGridData->Rows->Clear();
      dataGridData->Columns->Clear();
      Roman_ roman; Work_ work; auto v = work.Print();
      dataGridData->RowCount = roman.GetCount();
      if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
      else
      {
            Headers();
            dataGridData->AutoResizeRows();
            ShowRoman();
      }
}

System::Void       CurseProject::LibraryForm::Rasskaz_Click(System::Object^       sender,
System::EventArgs^ e)
{
      dataGridData->Rows->Clear();
      dataGridData->Columns->Clear();
      Rasskaz_ rasskaz; Work_ work; auto v = work.Print();
      dataGridData->RowCount = rasskaz.GetCount();
      if (v[0] == "") MessageBox::Show("Библиотека закрыта, ведется уборка помещения",
"Ошибка");
```

```
	else
	{
		Headers();
		dataGridData->AutoResizeRows();
		ShowRasskaz();
	}
}

System::Void	CurseProject::LibraryForm::butfav_Click(System::Object^	sender,
System::EventArgs^ e)
{
	Favourites_ favourites; auto valuefavourites_temp = favourites.GetFV();
	if (valuefavourites_temp[NULL] == "") { MessageBox::Show("Раздел Избранное пуст",
"Ошибка"); }
	else
	{

		dataGridData->Rows->Clear();
		dataGridData->Columns->Clear();
		dataGridData->RowCount = favourites.GetFW();
		Headers_B();
		ShowFav();
	}
}

System::Void	CurseProject::LibraryForm::Fav_Click(System::Object^	sender,
System::EventArgs^ e)
{
	Favourites_ fav; Work_ work; vector<string> vector_books;
	String^ temp_current_Cell;
	int ask; work.book();
	vector_books = work.Print();
	////
	if (vector_books.empty())
	{
		dataGridData->Rows->Clear();
		dataGridData->Columns->Clear();
	}
	else {
		work.book();
		temp_current_Cell = dataGridData->CurrentCell->Value->ToString();
		try {
			ask = Convert::ToInt16(temp_current_Cell);
```

```
                  if ((ask > work.GetCount()) || (vector_books[NULL] == "")) {
MessageBox::Show("Индекс не принадлежит диапазону", "Ошибка"); }
                  else {
                        work.Favourites(ask);
                        dataGridData->Rows->Clear();
                        dataGridData->Columns->Clear();
                        dataGridData->RowCount = work.GetCount();
                        vector_books = work.Print();
                        if (!vector_books.empty()) {
                              Headers();
                              Show();
                        }
                        else
                        {
                              dataGridData->Rows->Clear();
                              dataGridData->Columns->Clear();
                        }
                  }
            }
            catch (Exception^ e)
            {
                  MessageBox::Show("Выбран неверный индекс", "Ошибка...");
            }
      }
}

System::Void       CurseProject::LibraryForm::button2_Click(System::Object^       sender,
System::EventArgs^ e)
{
      Favourites_ favourites_name;
      auto favourites_temp_name = favourites_name.GetFV();
      if (favourites_temp_name[NULL] == "") MessageBox::Show("Вы не выбрали книги",
"Ошибка");
      else
      {
            srand(time(NULL));
            MessageBox::Show("Книги добавлены в Ваш профиль","Уведомление");
            int day = 1 + rand() % 30;
            this->temp = gcnew Temp();
            this->temp->order_id = Guid::NewGuid();
            this->temp->value = day;

            Task<System::Guid>^ thread = gcnew Task<System::Guid>(gcnew Func<Guid>(temp,
&Temp::D));
```

```
            thread->ContinueWith(gcnew Action<Task<Guid>^>(temp, &Temp::B));
            thread->Start();


      fstream File(FILE_FAVOURITES_NAME, ios::out);
            DateTime date1 = DateTime::Today;
            DateTime answer = date1.AddDays(day);
            int day_temp = Convert::ToInt16(date1.Day) + day;
            String^ Str = answer.ToString("m") + " <= дата последнего дня аренды книг ";
            MessageBox::Show(Str, "Успешно");
            dataGridData->Rows->Clear();
            dataGridData->Columns->Clear();
      }
}

System::Void       CurseProject::LibraryForm::button1_Click(System::Object^       sender,
System::EventArgs^ e)
{
      Form::Hide();
      LoginForm^ form = gcnew LoginForm();
      form->Show();
}
void CurseProject::LibraryForm::Headers()
{
      HeaderA();
      HeaderB();
      HeaderC();
      HeaderD();
      HeaderE();
      HeaderF();
}
void CurseProject::LibraryForm::HeaderA_busk()
{
      dataGridData->TopLeftHeaderCell->Value = "########";
      DataGridViewTextBoxColumn^ c1 = gcnew DataGridViewTextBoxColumn();
      c1->Name = "Список";
      c1->HeaderText = "Жанр";
      c1->Width = 150;
      dataGridData->Columns->Add(c1);

      dataGridData->AutoResizeColumn(0);
}
//new
void CurseProject::LibraryForm::HeaderA()
```

```
{
	DataGridViewTextBoxColumn^ c2 = gcnew DataGridViewTextBoxColumn();
	c2->Name = "Список";
	c2->HeaderText = "Жанр";
	c2->Width = 150;
	dataGridData->Columns->Add(c2);

	dataGridData->AutoResizeColumn(0);
}
void CurseProject::LibraryForm::HeaderB()
{
	DataGridViewTextBoxColumn^ c2 = gcnew DataGridViewTextBoxColumn();
	c2->Name = "Список";
	c2->HeaderText = "Название";
	c2->Width = 150;
	dataGridData->Columns->Add(c2);

	dataGridData->AutoResizeColumn(0);
}
void CurseProject::LibraryForm::HeaderC()
{
	DataGridViewTextBoxColumn^ c5 = gcnew DataGridViewTextBoxColumn();
	c5->Name = "Список";
	c5->HeaderText = "Год издания";
	c5->Width = 150;
	dataGridData->Columns->Add(c5);

	dataGridData->AutoResizeColumn(0);
}
void CurseProject::LibraryForm::HeaderD()
{
	DataGridViewTextBoxColumn^ c3 = gcnew DataGridViewTextBoxColumn();
	c3->Name = "Список";
	c3->HeaderText = "Автор";
	c3->Width = 150;
	dataGridData->Columns->Add(c3);

	dataGridData->AutoResizeColumn(0);
}

void CurseProject::LibraryForm::HeaderE()
{
	DataGridViewTextBoxColumn^ c4 = gcnew DataGridViewTextBoxColumn();
	c4->Name = "Список";
```

```
	c4->HeaderText = "Страницы";
	c4->Width = 150;
	dataGridData->Columns->Add(c4);

	dataGridData->AutoResizeColumn(0);
}

System::Void	CurseProject::LibraryForm::button3_Click(System::Object^	sender,
System::EventArgs^ e)
{
	if (temp->value == 0) MessageBox::Show("Вы не получали книги", "Ошибка");
	else	MessageBox::Show("До	даты	последнего	дня	аренды	книг:	"	+	this->temp-
>value.ToString() + " дней", "Статус");
}

void CurseProject::LibraryForm::ShowRasskaz()	// отображение ввиде таблицы
{
	int temp = 0;
	Rasskaz_ rasskaz;
	Work_ work;
	work.book();
	dataGridData->ClearSelection();
	smatch find_world;
	regex regular(SEARCH_RASSKAZ);
	std::vector<string> v = work.Print();
	vector<Heads> heads_v = DivideRow(v, work.GetCount());
	for (int i = rasskaz.GetBegin(); i < work.GetCount(); i++)
	{
		if (regex_search(v[i], find_world, regular)) {
			dataGridData->Rows[temp]->HeaderCell->Value = "=>";
			dataGridData->Columns[0]->HeaderCell->Value = "id";
			dataGridData->Rows[temp]->Cells[NULL]->Value	=	Convert::ToString(i	+
1);
			dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value		=
Convert_string_To_String(heads_v[i].Janr);
			dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value	=
Convert_string_To_String(heads_v[i].Nazvanie);
			dataGridData->Rows[temp]->Cells[Head_name::Year]->Value		=
Convert_string_To_String(heads_v[i].Year);
			dataGridData->Rows[temp]->Cells[Head_name::Author]->Value	=
Convert_string_To_String(heads_v[i].Author);
			dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value		=
Convert_string_To_String(heads_v[i].Pages) + " стр.";
```

```
                    dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value            =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
                    dataGridData->AutoResizeColumn(0);
                    dataGridData->AutoResizeRows();
                    temp++;
             }
      }
      dataGridData->AutoResizeColumn(0);
      dataGridData->AutoResizeRows();
}

void CurseProject::LibraryForm::Headers_B()
{
      HeaderA_busk();
      HeaderB();
      HeaderC();
      HeaderD();
      HeaderE();
}
void CurseProject::LibraryForm::HeaderF()
{
      DataGridViewTextBoxColumn^ c6 = gcnew DataGridViewTextBoxColumn();
      c6->Name = "Список";
      c6->HeaderText = "Экземпляры";
      c6->Width = 150;
      dataGridData->Columns->Add(c6);

      dataGridData->AutoResizeColumn(0);
}
void CurseProject::LibraryForm::ShowFav()
{
      int temp = 0;
      Favourites_ favourites;
      std::vector<string> v = favourites.GetFV();
      dataGridData->ClearSelection();
      vector<Heads> ones_v = DivideRow(v, favourites.GetFW());
      for (int i = 0; i < favourites.GetFW(); i++)
      {
            dataGridData->Rows[temp]->HeaderCell->Value = Convert::ToString(i + 1) + ".";
            dataGridData->Columns[0]->HeaderCell->Value = "Избранное";
            dataGridData->Rows[temp]->Cells[NULL]->Value = "###";
            dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value                    =
Convert_string_To_String(ones_v[i].Janr);
```

```
            dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value                =
Convert_string_To_String(ones_v[i].Nazvanie);
            dataGridData->Rows[temp]->Cells[Head_name::Year]->Value                    =
Convert_string_To_String(ones_v[i].Year);
            dataGridData->Rows[temp]->Cells[Head_name::Author]->Value                  =
Convert_string_To_String(ones_v[i].Author);
            dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value                   =
Convert_string_To_String(ones_v[i].Pages) + " стр.";
            dataGridData->AutoResizeColumn(0);
            dataGridData->AutoResizeRows();
            temp++;
      }
}

void CurseProject::LibraryForm::ShowRoman()
{
      int temp = 0;
      Roman_ roman;
      Work_ work;
      work.book();
      smatch find_world;
      regex regular(SEARCH_ROMAN);
      std::vector<string> v = work.Print();
      vector<Heads> heads_v = DivideRow(v, work.GetCount());
      for (int i = roman.GetBegin(); i < work.GetCount(); i++)
      {
            if (regex_search(v[i], find_world, regular)) {
                  dataGridData->Rows[temp]->HeaderCell->Value = "=>";
                  dataGridData->Columns[0]->HeaderCell->Value = "id";
                  dataGridData->Rows[temp]->Cells[NULL]->Value  =  Convert::ToString(i  +
1);
                  dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value              =
Convert_string_To_String(heads_v[i].Janr);
                  dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value          =
Convert_string_To_String(heads_v[i].Nazvanie);
                  dataGridData->Rows[temp]->Cells[Head_name::Year]->Value              =
Convert_string_To_String(heads_v[i].Year);
                  dataGridData->Rows[temp]->Cells[Head_name::Author]->Value            =
Convert_string_To_String(heads_v[i].Author);
                  dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value              =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
                  dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value            =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
                  dataGridData->AutoResizeColumn(0);
```

```
                dataGridData->AutoResizeRows();
                temp++;
            }
        }
        dataGridData->AutoResizeColumn(0);
        dataGridData->AutoResizeRows();
}

void CurseProject::LibraryForm::ShowPovest()
{
        int temp = 0;
        Povest_ povest;
        povest.Print();
        Work_ work;
        work.book();
        std::vector<string> v = work.Print();
        dataGridData->ClearSelection();
        smatch find_world;
        regex regular(SEARCH_POVEST);
        vector<Heads> heads_v = DivideRow(v, work.GetCount());
        for (int i = povest.GetBegin(); i < work.GetCount(); i++)
        {
            if (regex_search(v[i], find_world, regular)) {
                dataGridData->Rows[temp]->HeaderCell->Value = "=>";
                dataGridData->Columns[0]->HeaderCell->Value = "id";
                dataGridData->Rows[temp]->Cells[NULL]->Value  =  Convert::ToString(i  +
1);
                dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value              =
Convert_string_To_String(heads_v[i].Janr);
                dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value          =
Convert_string_To_String(heads_v[i].Nazvanie);
                dataGridData->Rows[temp]->Cells[Head_name::Year]->Value              =
Convert_string_To_String(heads_v[i].Year);
                dataGridData->Rows[temp]->Cells[Head_name::Author]->Value            =
Convert_string_To_String(heads_v[i].Author);
                dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value             =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
                dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value           =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
                dataGridData->AutoResizeColumn(0);
                dataGridData->AutoResizeRows();
                temp++;
            }
        }
```

```
	dataGridData->AutoResizeColumn(0);
	dataGridData->AutoResizeRows();
}

void CurseProject::LibraryForm::Show()
{
	Work_ work;
	work.book();
	auto v = work.Print();
	vector<Heads> heads_v = DivideRow(v, work.GetCount());
	for (int i = 0; i < work.GetCount(); i++)
	{
		dataGridData->Rows[i]->HeaderCell->Value = "=>";
		dataGridData->Columns[0]->HeaderCell->Value = "id";
		dataGridData->Rows[i]->Cells[NULL]->Value = Convert::ToString(i + 1);
		dataGridData->Rows[i]->Cells[Head_name::Janr]->Value                    =
Convert_string_To_String(heads_v[i].Janr);
		dataGridData->Rows[i]->Cells[Head_name::Nazvanie]->Value                =
Convert_string_To_String(heads_v[i].Nazvanie);
		dataGridData->Rows[i]->Cells[Head_name::Year]->Value                    =
Convert_string_To_String(heads_v[i].Year);
		dataGridData->Rows[i]->Cells[Head_name::Author]->Value                  =
Convert_string_To_String(heads_v[i].Author);
		dataGridData->Rows[i]->Cells[Head_name::Pages]->Value                   =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
		dataGridData->Rows[i]->Cells[Head_name::Ekzemps]->Value                 =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
		dataGridData->AutoResizeColumn(0);
		dataGridData->AutoResizeRows();
	}
}
void CurseProject::LibraryForm::ShowStih()
{
	int temp = 0;
	Stih_ stih;
	stih.Print();
	Work_ work;
	work.book();
	std::vector<string> v = work.Print();
	dataGridData->ClearSelection();
	smatch find_world;
	regex regular(SEARCH_STIH);
	vector<Heads> heads_v = DivideRow(v, work.GetCount());
	for (int i = stih.GetBegin(); i < work.GetCount(); i++)
```

```
	{
		if (regex_search(v[i], find_world, regular)) {
			dataGridData->Rows[temp]->HeaderCell->Value = "=>";
			dataGridData->Columns[0]->HeaderCell->Value = "id";
			dataGridData->Rows[temp]->Cells[NULL]->Value  =  Convert::ToString(i  +
1);
			dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value              =
Convert_string_To_String(heads_v[i].Janr);
			dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value          =
Convert_string_To_String(heads_v[i].Nazvanie);
			dataGridData->Rows[temp]->Cells[Head_name::Year]->Value              =
Convert_string_To_String(heads_v[i].Year);
			dataGridData->Rows[temp]->Cells[Head_name::Author]->Value            =
Convert_string_To_String(heads_v[i].Author);
			dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value              =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
			dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value            =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
			dataGridData->AutoResizeColumn(0);
			dataGridData->AutoResizeRows();
			temp++;
		}
	}
	dataGridData->AutoResizeColumn(0);
	dataGridData->AutoResizeRows();
}
void CurseProject::LibraryForm::ShowSkazka()
{
	int temp = 0;
	Skazka_ skazka;
	skazka.Print();
	Work_ work;
	work.book();
	std::vector<string> v = work.Print();
	dataGridData->ClearSelection();
	smatch find_world;
	regex regular(SEARCH_SKAZKA);
	vector<Heads> heads_v = DivideRow(v, work.GetCount());
	for (int i = skazka.GetBegin(); i < work.GetCount(); i++)
	{
		if (regex_search(v[i], find_world, regular)) {
			dataGridData->Rows[temp]->HeaderCell->Value = "=>";
			dataGridData->Columns[0]->HeaderCell->Value = "id";
```

```
                    dataGridData->Rows[temp]->Cells[NULL]->Value  =  Convert::ToString(i  +
1);
                    dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value              =
Convert_string_To_String(heads_v[i].Janr);
                    dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value          =
Convert_string_To_String(heads_v[i].Nazvanie);
                    dataGridData->Rows[temp]->Cells[Head_name::Year]->Value              =
Convert_string_To_String(heads_v[i].Year);
                    dataGridData->Rows[temp]->Cells[Head_name::Author]->Value            =
Convert_string_To_String(heads_v[i].Author);
                    dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value             =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
                    dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value           =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
                    dataGridData->AutoResizeColumn(0);
                    dataGridData->AutoResizeRows();
                    temp++;
             }
      }
      dataGridData->AutoResizeColumn(0);
      dataGridData->AutoResizeRows();
}
void CurseProject::LibraryForm::ShowPiesa()
{
      int temp = 0;
      Piesa_ piesa;
      piesa.Print();
      Work_ work;
      work.book();
      std::vector<string> v = work.Print();
      dataGridData->ClearSelection();
      smatch find_world;
      regex regular(SEARCH_PIESA);
      vector<Heads> heads_v = DivideRow(v, work.GetCount());
      for (int i = piesa.GetBegin(); i < work.GetCount(); i++)
      {
             if (regex_search(v[i], find_world, regular)) {
                    dataGridData->Rows[temp]->HeaderCell->Value = "=>";
                    dataGridData->Columns[0]->HeaderCell->Value = "id";
                    dataGridData->Rows[temp]->Cells[NULL]->Value  =  Convert::ToString(i  +
1);
                    dataGridData->Rows[temp]->Cells[Head_name::Janr]->Value              =
Convert_string_To_String(heads_v[i].Janr);
```

```
                    dataGridData->Rows[temp]->Cells[Head_name::Nazvanie]->Value          =
Convert_string_To_String(heads_v[i].Nazvanie);
                    dataGridData->Rows[temp]->Cells[Head_name::Year]->Value              =
Convert_string_To_String(heads_v[i].Year);
                    dataGridData->Rows[temp]->Cells[Head_name::Author]->Value            =
Convert_string_To_String(heads_v[i].Author);
                    dataGridData->Rows[temp]->Cells[Head_name::Pages]->Value              =
Convert_string_To_String(heads_v[i].Pages) + " стр.";
                    dataGridData->Rows[temp]->Cells[Head_name::Ekzemps]->Value            =
Convert_string_To_String(heads_v[i].Ekzemps) + " экз.";
                    dataGridData->AutoResizeColumn(0);
                    dataGridData->AutoResizeRows();
                    temp++;
              }
       }
       dataGridData->AutoResizeColumn(0);
       dataGridData->AutoResizeRows();
}
```

LibrarianForm1.cpp

```
#include "LibrarianForm1.h"
#include "Functions.h"
#include "Classes.h"

using namespace System;

System::Void CurseProject::LibrarianForm1::button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
       String^ str = "Povest ";
       str += textBox8->Text->ToString() + " "
              + numericUpDown2->Text->ToString() + " "
              + textBox6->Text->ToString() + " "
              + numericUpDown1->Text->ToString() + " "
              + numericUpDown3->Text->ToString();
       std::string temp = Stos(str);
       std::ofstream File(MAIN_BOOKS, std::ios::app);
       File << "\n" << temp;
       MessageBox::Show("Успешно", "Состояние");
       File.close();
       this->Close();
}
```

LibrarianForm.cpp

```
#include "LibrarianForm.h"
#include "LoginForm.h"
#include "Functions.h"
#include "Classes.h"
#include "LibrarianForm1.h"
#include "LibrarianForm2.h"
#include "LibrarianForm3.h"
```

```
#include "LibrarianForm4.h"
#include "LibrarianForm5.h"
#include "LibrarianForm6.h"

using namespace System;
System::Void CurseProject::LibrarianForm::ShowL_Click(System::Object^ sender,
System::EventArgs^ e)
{
	Show_Librarian_retry();
}

void CurseProject::LibrarianForm::ShowLibrarian()
{
	Librarian lib;
	std::vector<std::string> v;
	v = lib.Print_Lib_data();
	int temp = 0;
	dataGridView1->TopLeftHeaderCell->Value = "id";
	for (int i = 0; i < v.size(); i++)
	{
		dataGridView1->Rows[temp]->HeaderCell->Value = Convert::ToString(i + 1);
		dataGridView1->Columns[0]->HeaderCell->Value = "#";
		dataGridView1->Rows[temp]->Cells[0]->Value = "->";
		dataGridView1->Rows[temp]->Cells[1]->Value = Convert_string_To_String(v[i]);
		temp++;
	}
	dataGridView1->AutoResizeColumn(0);
	dataGridView1->AutoResizeRows();
}

Void CurseProject::LibrarianForm::HeaderLibrarian()
{
	DataGridViewTextBoxColumn^ c1 = gcnew DataGridViewTextBoxColumn();
	c1->Name = "Список";
	c1->HeaderText = "Librarian View";
	c1->Width = 150;
	dataGridView1->Columns->Add(c1);

	dataGridView1->AutoResizeColumn(0);
}

System::Void CurseProject::LibrarianForm::button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
	Form::Hide();
	LoginForm^ form = gcnew LoginForm();
	form->Show();
}

System::Void CurseProject::LibrarianForm::Dat_Click(System::Object^ sender,
System::EventArgs^ e)
{
	MessageBox::Show("Успешно", "Состояние");
	TransferData();
}

System::Void CurseProject::LibrarianForm::Povest_Click(System::Object^ sender,
System::EventArgs^ e)
{
	LibrarianForm1^ form1 = gcnew LibrarianForm1;
	form1->ShowDialog();
}
System::Void CurseProject::LibrarianForm::Stih_Click(System::Object^ sender,
System::EventArgs^ e)
{
```

```
	LibrarianForm2^ form2 = gcnew LibrarianForm2;
	form2->ShowDialog();
}
System::Void CurseProject::LibrarianForm::Roman_Click(System::Object^ sender,
System::EventArgs^ e)
{
	LibrarianForm3^ form3 = gcnew LibrarianForm3;
	form3->ShowDialog();
}

System::Void CurseProject::LibrarianForm::Rasskaz_Click(System::Object^ sender,
System::EventArgs^ e)
{
	LibrarianForm4^ form4 = gcnew LibrarianForm4;
	form4->ShowDialog();
}
System::Void CurseProject::LibrarianForm::Skazka_Click(System::Object^ sender,
System::EventArgs^ e)
{
	LibrarianForm5^ form5 = gcnew LibrarianForm5;
	form5->ShowDialog();
}

System::Void CurseProject::LibrarianForm::Piesa_Click(System::Object^ sender,
System::EventArgs^ e)
{
	LibrarianForm6^ form6 = gcnew LibrarianForm6;
	form6->ShowDialog();
}

void CurseProject::LibrarianForm::Show_Librarian_retry()
{
	Librarian lib;
	dataGridView1->Rows->Clear();
	dataGridView1->Columns->Clear();
	dataGridView1->RowCount = lib.GetCount();

	HeaderLibrarian();
	ShowLibrarian();

}
```

LibrarianForm2.cpp

```
#include "LibrarianForm2.h"
#include "Functions.h"
#include "Classes.h"
System::Void CurseProject::LibrarianForm2::button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
	String^ str = "Stihotvorenie ";
	str += textBox8->Text->ToString() + " "
		+ numericUpDown2->Text->ToString() + " "
		+ textBox6->Text->ToString() + " "
		+ numericUpDown1->Text->ToString() + " "
		+ numericUpDown3->Text->ToString();
	std::string temp = Stos(str);
	std::ofstream File(MAIN_BOOKS, std::ios::app);
	File << "\n" << temp;
	MessageBox::Show("Успешно", "Состояние");
	File.close();
	this->Close();
}
```

LibrarianForm3.cpp

```
#include "LibrarianForm3.h"
#include "Functions.h"
#include "Classes.h"
System::Void CurseProject::LibrarianForm3::button2_Click(System::Object^ sender,
System::EventArgs^ e)
{
	String^ str = "Roman ";
	str += textBox1->Text->ToString() + " "
		+ numericUpDown2->Text->ToString() + " "
		+ textBox4->Text->ToString() + " "
		+ numericUpDown1->Text->ToString() + " "
		+ numericCount->Text->ToString();
	std::string temp = Stos(str);
	std::ofstream File(MAIN_BOOKS, std::ios::app);
	File << "\n" << temp;
	MessageBox::Show("Успешно", "Состояние");
	File.close();
	this->Close();
}
```

LibrarianForm4.cpp

```
#include "LibrarianForm4.h"
#include "Functions.h"
#include "Classes.h"
System::Void CurseProject::LibrarianForm4::button2_Click(System::Object^ sender,
System::EventArgs^ e)
{
	String^ str = "Rasskaz ";
	str += textBox1->Text->ToString() + " "
		+ numericUpDown2->Text->ToString() + " "
		+ textBox4->Text->ToString() + " "
		+ numericUpDown1->Text->ToString() + " "
		+ numericCount->Text->ToString();
	std::string temp = Stos(str);
	std::ofstream File(MAIN_BOOKS, std::ios::app);
	File << "\n" << temp;
	MessageBox::Show("Успешно", "Состояние");
	File.close();
	this->Close();
}
```

LibrarianForm5.cpp

```
#include "LibrarianForm5.h"
#include "Functions.h"
#include "Classes.h"
System::Void CurseProject::LibrarianForm5::button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
	String^ str = "Skazka ";
	str += textBox8->Text->ToString() + " "
		+ numericUpDown2->Text->ToString() + " "
		+ textBox6->Text->ToString() + " "
		+ numericUpDown1->Text->ToString() + " "
		+ numericUpDown3->Text->ToString();
	std::string temp = Stos(str);
	std::ofstream File(MAIN_BOOKS, std::ios::app);
	File << "\n" << temp;
```

```
	MessageBox::Show("Успешно", "Состояние");
	File.close();
	this->Close();
}
```

LibrarianForm6.cpp

```
#include "LibrarianForm6.h"
#include "Functions.h"
#include "Classes.h"
System::Void CurseProject::LibrarianForm6::button1_Click(System::Object^ sender,
System::EventArgs^ e)
{
	String^ str = "Piesa ";
	str += textBox8->Text->ToString() + " "
		+ numericUpDown2->Text->ToString() + " "
		+ textBox6->Text->ToString() + " "
		+ numericUpDown1->Text->ToString() + " "
		+ numericUpDown3->Text->ToString();
	std::string temp = Stos(str);
	std::ofstream File(MAIN_BOOKS, std::ios::app);
	File << "\n" << temp;
	MessageBox::Show("Успешно", "Состояние");
	File.close();
	this->Close();
}
```