|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА ИУ8 ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К КУРСОВОЙ РАБОТЕ***

***НА ТЕМУ:***

**Система для учёта и поиска книг в библиотеке**

Студент ИУ8 – 32 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. В. Борисов\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель курсовой работы **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Колесников**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Колесников**

(Подпись, дата) (И.О.Фамилия)

*2018г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой ИУ8

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Басараб

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине: Технологии и методы программирования

Студент группы: ИУ8 – 32

Борисов Сергей Викторович

(Фамилия, имя, отчество)

Тема курсовой работы: Система для учёта и поиска книг в библиотеке\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к 3 нед., 50% к 9 нед., 75% к 12 нед., 100% к 15 нед.

***Задание:*** Разработать программу, позволяющую вести электронный учёт и поиск книг в библиотеке с использованием баз данных. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Оформление курсовой работы:***

Расчетно-пояснительная записка на \_\_\_\_\_ листах формата А4.

Перечень графического (иллюстрированного) материала (презентация, слайды).

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **А.В. Колесников**

(Подпись, дата) (И.О.Фамилия)

**Студент** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **С.В. Борисов**

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**Тема курсовой работы**: Система для учёта и поиска книг в библиотеке.

**Цель работы**: разработать программу, реализующую автоматизированный электронный учёт, поиск, выдачу и возврат книг в библиотечном фонде с использованием баз данных. Программа должна иметь графический интерфейс, обеспечивающий достаточно быстрый и удобный поиск, выдачу и возврат нужных книг для читателя, а также должна вести автоматизированный учёт лиц, имеющих задолженность перед библиотекой.

**Задачи работы:**

1. Реализовать графический интерфейсы программы.
2. Выбрать СУБД, наиболее подходящую для выполнения поставленной цели работы.
3. Построить систему баз данных, отвечающую поставленной задаче.
4. Реализовать функцию для добавления книг в фонд.
5. Реализовать функцию удаления книг из фонда.
6. Реализовать функции для создания заказа.
7. Реализовать функцию для завершения заказа.
8. Реализовать автоматизированный учёт должников.
9. Реализовать функцию поиска книг в фонде.
10. Реализовать регистрацию читателя в базе данных библиотеки.
11. Реализовать ввод и проверку пользовательских данных (пароль, логин) для подключения к БД фонда.

**Глоссарий**

**СУБД** – система управления базами данных.

**SQL** – язык структурированных запросов для управления соответствующей базой данных.

**Коннектор** (connector) – часть программного обеспечения, целью которого является «стыковка» различных технологий между собой. Коннектор позволяет двум или более технологиям взаимодействовать между собой, выполняя необходимые процедуры и передавая данные.

**GNU** (General Public License – Универсальная общественная лицензия GNU, Универсальная общедоступная лицензия GNU или Открытое лицензионное соглашение GNU) — [лицензия](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F) на [свободное программное обеспечение](https://ru.wikipedia.org/wiki/%D0%A1%D0%B2%D0%BE%D0%B1%D0%BE%D0%B4%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), созданная в рамках проекта [GNU](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82_GNU) в [1988](https://ru.wikipedia.org/wiki/1988) г., по которой [автор](https://ru.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80) передаёт [программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) в  [общественную собственность](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%BE%D0%B1%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

**Модальное окно** в [графическом интерфейсе пользователя](https://ru.wikipedia.org/wiki/GUI) — [окно](https://ru.wikipedia.org/wiki/%D0%9E%D0%BA%D0%BD%D0%BE_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которое блокирует работу пользователя с родительским приложением до тех пор, пока пользователь это окно не закроет.

**Ресурсы** — это элементы интерфейса, предоставляющие информацию пользователю. К ресурсам относятся точечные рисунки, значки, панели инструментов и курсоры. Некоторые ресурсы можно использовать для выполнения действия, такого как выбор в меню или ввод данных в диалоговом окне.

**БД** – база данных.

**Введение**

Внедрение автоматизации в библиотеку должно повысить производительность и качество труда работников библиотеки. Процесс автоматизации деятельности библиотекаря должен обеспечить  быстрый поиск, выдачу и возврата книг, а также вести автоматизированный учет должников, заказ литературы, поиск необходимой читателю литературы с использованием  электронных каталогов библиотеки. Таким образом, целью решения задачи автоматизации является сокращение времени, затрачиваемого библиотекарем на поиск книг в каталоге и оформление документов, связанных с выдачей и возвратом книг, снижение до минимума количества ошибок, допускаемых при заполнении документов и оптимизация сбора библиотечной статистики.

**Архитектура**

Хранением большого массива данных, содержащего в себе информацию о библиотечном фонде, должна заниматься база данных, находящаяся на сервере в локальной сети библиотеки. Компьютеры операторов библиотеки должны будут только выводить запрошенную у Базы Данных информацию и отправлять запросы обратно. Всю работу по обработке запросов, присланных клиентским приложением, должно будет заниматься Ядро СУБД. Таким образом, компьютеры, на которых будет запущен клиент приложения, не будут сильно загружены и поэтому не должны быть мощными.

Благодаря широкому распространению ОС Windows и умению большинства офисных работников использовать данную ОС было решено разрабатывать программу под ОС Windows. Для написания графического интерфейса использована библиотека, разработанная Microsoft специально для Windows – MFC.Библиотека, облегчает работу с GUI путём создания каркаса приложения — «скелетной» программы, автоматически создаваемой по заданному макету интерфейса и полностью берущей на себя рутинные действия по его обслуживанию (отработка оконных событий, пересылка данных между внутренними буферами элементов и переменными программы и т. п.). MFC не что иное как оболочка WinApi  для языка С++, следовательно интерфейс приложения будет быстрым и не требовательным к ресурсам, что положительно скажется на использовании программы в работе + Microsoft продолжают поддержку данной библиотеки и сегодня.

Для управления базами данных, хранящих информацию о фонде и о читателях библиотеки использована СУБД MySQL. MySQL можно использовать в качестве внешнего и внутреннего сервера. Данная СУБД является хорошим решением для малых и средних приложений, а также распространяется под GNU лицензией. MySQL обладает высокой производительностью и достаточно надёжен, а также не требует больших ресурсов от сервера+ для связывания клиентской оболочки с СУБД существует специальный C++ коннектор, который позволяет посылать запросы к СУБД. По этим причинам в качестве СУБД был выбран MySQL.

**Разработка программы**

**Задача 1.**

Графический интерфейс программы будет состоять из нескольких окон, вызываемых программно или по желанию пользователя, после нажатия на соответствующие кнопки.

Главное окно состоит из:

1. Таблицы списка книг, находящихся в фонде библиотеки.
2. Таблицы читателей, взявших определенную книгу по заказу.
3. Таблицы должников, с отображением количества дней прострочки аренды.
4. Версии используемого клиента сервера MySQL (левый нижний угол).
5. Кнопки Совершить заказ.
6. Кнопки Завершить заказ.
7. Кнопки Добавить/Удалить книгу.
8. Кнопок ОК и Выйти.

MFC содержит большую иерархию классов, написанных на С++, на её вершине расположился класс CObject, который содержит различные функции, используемые во время выполнения программы. От него наследуется класс CCmdTarget,предназначенный для обработки сообщений, а уже от него наследуется один из важнейших классов CWnd. CWnd является базовым для создания всех окон, включая диалоговые окна, которые будут основой интерфейса программы. Ещё от класса CCmdTarget наследуется СWinThread, а от него порождается класс CWinApp, предназначенный для создания самого приложения (диалога). В программе может иметься только один экземпляр этого класса, сразу после его создания начинается выполнение программы. Любой диалог будет иметь свой класс, который наследуется от класса диалога CDialogEx, который в свою очередь наследуется от базового диалога CDialog, а тот наследуется от CWnd. Каждый модальный диалог будет иметь свой обработчик событий Windows.

У всех классов окон есть виртуальная функция virtual BOOL OnInitDialog() (в которой необходимо объявить её же только от CDialogEx, так как класс родительский). Эта функция будет вызываться при каждом запуске модального диалога, инициализируя необходимые поля, приTRUE, если вы не устанавливаете фокус ввода на элемент управления в диалоговом окне, в противном случае FALSE, если вы хотите установить фокус на конкретный элемент управления.

Каждое диалоговое окно имеет свою карту событий BEGIN\_MESSAGE\_MAP(CDialogCustom,CDialogEx), где первый аргумент это класс созданного нами диалога, 2-ой параметр это его родитель. Любое сообщение, явно обрабатываемое в программе, нужно связать с его обработчиком,например,ON\_BN\_CLICKED(IDCANCEL,&CBookingAccountingDlg::OnBnClickedCancel) - сообщение посланное от кнопки отмена связывается с функцией afx\_msg void OnBnClickedCancel(), (afx\_msg - спецификатор объявления обработчика сообщения)которая вызовется при обработке данного сообщения.

В конструкторе создаваемого диалога необходимо вызвать конструктор класса родителя CDialogExи задать начальные значения текстовых полей редактирования. В случае с главным окном можно загрузить иконку приложения из ресурсовm\_hIcon = AfxGetApp() -> LoadIcon(IDR\_MAINFRAME) AfxGetApp() – возвращает указатель на себя.

Метод DoDataExchange применяется для связывания с органами управления диалоговой панели элементов управляющего ей класса.

Чтобы проверить, что индетификатор IDM\_ABOUTBOX относится к системным программам

ASSERT((IDM\_ABOUTBOX & 0xFFF0) == IDM\_ABOUTBOX);

ASSERT(IDM\_ABOUTBOX < 0xF000);

afx\_msg OnSysCommand(UINT nID, LPARAM lParam) работает с системными вызовами, такими как изменение размеров диалогового окна и его сворачивание.

Это скелет всех модальных диалоговых окон, которые будут создаваться и вызываться в программе.

**Задача 2.** Оптимальное СУБД для данной задачи это MySQL (см. пункт Архитектура)

**Задача 3.** Для эффективного выполнения поставленной задачи было решено создать 3 таблицы.

1. Таблица, содержащая информацию о книгах фонда и имеющая следующие столбцы: индивидуальный номер книги, название, автор, жанр, издательство, стоимость, год издания, стеллаж, номер места на стеллаже, количество, штрих-код. Для поддержки кириллицы в таблице кодировка изменена на cp1251. Для корректного хранения информации для всех столбцов включён модификатор – не пустое значение. Для столбца id\_book включены модификаторы на автоинкрементирование, неотрицательность, на однозначное индентифицирование каждой записи (primary key), уникальность и не пустоту.

Для стоимости, года издания, стеллажа, номера места на стеллаже, количества, штрих-кода включён модификатор неотрицательность, что позволяет проверять эти значения не в программе клиента БД, а в СУБД. Ещё для штрих-кода выставлен параметр уникальности(если есть несколько книг одного вида, то штрих-коды у них одинаковы).

1. Таблица, содержащая информацию о читателях библиотеки, имеющая следующие столбцы: уникальный номер читателя, имя, фамилию, отчество, адрес проживания, моб. номер, номер паспорта, дата регистрации в базе. Для всех столбцов включён модификатор – не пустое значение. Для столбца id\_reader включены модификаторы на автоинкрементирование, неотрицательность, на однозначное индентифицирование каждой записи (primary key) уникальность и не пустоту. Для моб. телефона и номера паспорта установлен параметр на неотрицательность значений. Столбец даты регистрации имеет формат даты MySQL ГГ-ММ-ДД.
2. Таблица, содержащая информацию по прошлым и текущим заказам включает в себя следующие столбцы: уникальный номер заказа, уникальный номер читателя, уникальный номер книги, дату выдачи книги, количество дней на сколько выдали книгу и дату возврата книги.Для столбца id\_orderвключены модификаторы на автоинкрементирование, неотрицательность, на однозначное индентифицирование каждой записи (primary key), уникальность и не пустоту. Для всех остальных столбцов кроме receive\_date включён параметр не пустоты. Для всех столбцов кроме receive\_dateи give\_date включён параметр не отрицательности. Для удобства и дополнительной надёжности БД, осуществил связывание таблиц книги-заказы и читатели-заказы по значениям books.id\_book = order.id\_book и books.id\_reader = order.id\_reader. Эти связи называются один ко многим. Благодаря им нельзя будет добавить в таблицу заказов, те id которых нет в таблицах книг или читателей + выбрана связь CASCADE, что позволяет обновлять данные в дочерней таблице, если поменялись родительские значения и при удалении родительских id в таблице заказы удалить все связанные с этими id строки.

**Задача 4.**

Функция добавления книг в фонд описана в файле AddDelBookWnd.cpp

Запрос на добавление книги в БД будет выглядеть так:

INSERT INTO books ( title, autor, type, publisher, cost, year,rack,place\_rack, amount, barcode) VALUES (\"" + m\_TitleEdit + "\", \

\"" + m\_AutorEdit + "\" , \"" + m\_CategoryEdit + "\",\"" + m\_PublisherEdit + "\",\"" + m\_CostEdit + "\",\"" + m\_YearEdit + "\", \

\"" + m\_Rack+ "\",\"" + m\_PlaceRack + "\",\"" + m\_AmountEdit + "\",\"" + m\_BarcodeEdit + "\" )

Переменные начинающиеся с буквы m, считываются из контролов окна меню добавления книги.

Функция mysql\_query(coonection,query) возвратит значение не равное 0 в случае не выполнения составленного запроса. При неполном заполнении полей для книги СУБД вернёт сообщение об ошибке.

**Задача 5.**

При удалении книг из фонда, кроме проверки самой СУБД происходит собственная проверка на существование удаляемой книги по штрих-коду, с целью вывода ошибки на русском я зыке. Код удаления будет выглядеть таким образом.

CString query1 = "SELECT \* FROM books WHERE books.barcode = \"" + str + "\" ;";

if (mysql\_query(connection, query1) != 0) {

CString error\_str = mysql\_error(&mysql);

AfxMessageBox(error\_str, MB\_ICONSTOP);

return false;

}

result = mysql\_store\_result(connection);

if (mysql\_num\_rows(result) > 0) return true;

str -  переменная хранящая введённый пользователем штрих – код. Функция mysql\_num\_rows(result) – вернёт количество строк, содержащихся в ответе СУБД на запрос.

В случае полного удаления книги из фонда, запрос к СУБД будет выглядеть так:

DELETE FROM books WHERE books.barcode = \"" + m\_BarcodeEdit + "\" ;

Если нужно удалить не всё количество книг из фонда библиотеки, к примеру, пришла в негодность лишь некоторая часть книг, то запрос на изменение количества книг в БД:

UPDATE books SET books.amount = books.amount - \"" + m\_DelCount + "\" WHERE barcode = \"" + m\_BarcodeEdit + "\" ;

m\_DelCount – количество удаляемых книг,

m\_BarcodeEdit – штрих – код удаляемых книг.

Благодаря тому, что в базе данных поле amount хранит только не отрицательные значения, в случае m\_DelCount > amount, БД вернёт сообщение об ошибке.

Для того чтобы удалить книгу из БД, нужен лишь штрих – код, в случае полного удаления записи в таблице книги, нужно поставить выбрать данный пункт в диалоговом окне, если надо удалить лишь часть экземпляров надо ввести желаемое значение в соответствующие поле в диалоге.

**Задача 6, 9,10.**

Для соверешения заказа необходимо знать данные о читателе, о книге и о времени её аренды, которою он хочет взять.

Для удобства и экономии времени пользователю программы не нужно регистрировать читателе в отдельном окне. В случае если читатель не зарегестрирован в системе учёта, то его данные водятся прямо в окне совершения заказа. Происходит проверка верность ввода имени, фамилии и отчества (должно быть 3 разделённых пробелом слова, если меньше то программа предупердит об этом и не зарегистрирует чичтателя ). Код проверки на валидность ИФО.

//Добавление нового пользователя

if (!m\_FIOreader.IsEmpty() && !m\_AddresReader.IsEmpty() && !m\_trelnumberReader.IsEmpty() && !m\_PassportnumberReader.IsEmpty()) {

CStringArray arr;

int curPos = 0;

CString resToken = m\_FIOreader.Tokenize(\_T(" "), curPos);

while (resToken != \_T(""))

{

arr.Add(resToken);

resToken = m\_FIOreader.Tokenize(\_T(" "), curPos);

}

if (arr.GetCount() == 3) { //проверка на корректность ФИО

CString FirstName, SecondName, ThirdName;

FirstName = arr[0];

SecondName = arr[1];

ThirdName = arr[2];

В случае правильности ввода посылает запрос к MySQL:

INSERT INTO readers VALUES (NULL, \"" + FirstName + "\" , \"" + SecondName + "\" , \"" + ThirdName + "\" ,\

\"" + m\_AddresReader + "\" , \"" + m\_trelnumberReader + "\" , \"" + m\_PassportnumberReader + "\" , \"" + m\_StartRegistration + "\" );

Первое значение NULL, так как id\_reader инкрементируется СУБД автоматически.

Поля, которые подразумевают ввод только цифр не принимают буквенные символы (поле для номера телефона и номера пасспорта). Изначально поля даты регистрации и даты совершения заказа заполнены текущей системной датой компьютера и заполнения не требуют.Если добавление новой записи в таблицу readers произошло без ошибок, то выведется сообщение с индивидуальным номером читателя, его можно использовать, например, как номер читательского билета.

if (correct\_order) {

CString query2 =''SELECT \* FROM readers WHERE pasport\_number = \"" + m\_PassportnumberReader + "\" ;''

mysql\_query(connection, query2);

result = mysql\_store\_result(connection);

row = mysql\_fetch\_row(result);

idReader = row[0];

CString FIO = row[1];

FIO += \_T(" ");

FIO += row[2];

FIO += \_T(" ");

FIO += row[3];

CString message = \_T("Пользователь ") + FIO + \_T(" был добавлен в БД с номером ") + idReader;

m\_NumberReader.SetWindowText(idReader); // подставить в поле номер читателя только-что добавленного читателя

AfxMessageBox(message, MB\_ICONINFORMATION);

}

Если системе не хватит данных о пользователе и о количестве дней аренды книги, будет вызвано информационное окно с информацией об ошибке.

Если пользователь уже зарегистрирован в системе, то используется его номер.

Чтобы найти нужную книгу для поиска доступно название книги, её автор, и год издания. Поиск осуществляется по этим 3-м значением и в таблицу выводятся все варианты в которых встретилась нужная подстрока. Для вывода всех вариантов, содержащих данную подстроку используется такой SQL запрос:

SELECT \* FROM books WHERE title LIKE \"%" + m\_NameBook + "%\" AND autor LIKE \"%" + m\_AutorBook + "%\" AND year LIKE \"%" + m\_YearBook + "%\" ;

Знаки процета в начале и в конце подстрок означают, что поиск происходит как с начала слова, так и с конца.

Все найденные альтернативы выводятся в таблицу. Пользователь выбирает нужную книгу и выделяет её. Затем происходит ряд SQL запросов:

1. Уменьшение количества выбранной книги на 1.

UPDATE books SET books.amount = books.amount - 1 WHERE books.id\_book = \"" + id\_book + "\" ;

1. Добавление заказа в таблицу mydb.order со всеми нужными переменными:

INSERT INTO mydb.order VALUES (NULL, \"" + idReader + "\",\"" + id\_book + "\", \"" + m\_StartTrial + "\", \"" + m\_DaysTrial + "\", NULL );

Первая переменная NULL, так как происходит автоматическое инкреметрирование.

1. Затем в качестве дополнительной проверки берётся последняя запись в таблице заказов(т.е только что добавленная) и извлекается из неё номер заказа, для последующего использования при выдаче книги.

SELECT id\_order FROM mydb.order ORDER BY id\_order DESC LIMIT 1;

**Задача 7.**

Чтобы завершить заказ можно воспользоваться информацией о номере заказа, который хотим закрыть, или же в случае его утери и т.п, можно воспользоваться штрих-кодом книги, которую необходимо вернуть. При 1-ом случае информация о читателе не требуется, во 2-ом нужно знать либо номер читателя в базе либо его номер паспорта. Возникает 3 варианта завершения заказа.

1. Известен номер заказа.

Отправляем запрос на увеличение количества возвращаемой книги на 1.

UPDATE books b JOIN (SELECT \* FROM mydb.order o WHERE o.id\_order = \"" + m\_NumOrder + "\") AS temp ON b.id\_book = temp.id\_book SET b.amount = b.amount + 1;

Так как изменяемое значение в таблице books и сама таблице используется для поиска нужной строки, используем псевдонимы для таблиц books,order.Если запрос прошёл успешно добавляем в строку завершаемого заказа текущую дату, это будет сигнализировать о завершении заказа читателем.

UPDATE mydb.order SET mydb.order.receive\_date=\"" + m\_EndTrial + "\" WHERE mydb.order.id\_order = \"" + m\_NumOrder + "\" AND mydb.order.receive\_date IS NULL;

1. Известен штрих – код и номер читателя в базе.

Отправляем запрос на увеличение количества возвращаемой книги на 1.

UPDATE books SET books.amount = books.amount + 1 WHERE books.barcode = \"" + m\_BarCodeBook + "\";

Если запрос прошёл успешно добавляем в строку завершаемого заказа текущую дату.

UPDATE mydb.order o JOIN (SELECT o.id\_order FROM books b JOIN mydb.order o ON b.id\_book = o.id\_book WHERE b.barcode = \"" + m\_BarCodeBook + "\" AND o.id\_reader = \"" + m\_NumReader + "\" AND o.receive\_date IS NULL ORDER BY o.give\_date ASC LIMIT 1) AS temp \

ON o.id\_order = temp.id\_order SET o.receive\_date = \"" + m\_EndTrial + "\";

Данный запрос завершает самый ранний заказ, если читатель брал две одинаковых книги в разное время. Сначала выполняем вложенный запрос на объединение order & books с условием что id\_book одинаковы в обоих таблицах + помимо этого штрих – код должен совпасть с пользовательским как и номер пользователя с его id в order.

1. Известен штрих – код и паспорт читателя в базе.

UPDATE books SET books.amount = books.amount + 1 WHERE books.barcode = \"" + m\_BarCodeBook + "\";

Запрос на завершение заказа:

UPDATE mydb.order o JOIN (SELECT o.id\_order FROM mydb.order o JOIN books b ON b.id\_book = o.id\_book \

JOIN readers r ON o.id\_reader = r.id\_reader WHERE b.barcode =\"" + m\_BarCodeBook + "\" AND r.pasport\_number = \"" + m\_PasportReader + "\" AND o.receive\_date IS NULL \

ORDER BY o.give\_date ASC LIMIT 1) AS temp ON o.id\_order = temp.id\_order SET o.receive\_date =\"" + m\_EndTrial + "\";

Данный запрос завершает самый ранний заказ, если читатель брал две одинаковых книги в разное время.

В подзапросе содержится 2 запроса на объединение таблиц order & books (o.id\_book=b.id\_book) и объединение order & readers (o.id\_reader = r.id\_reader).Затем отсеиваем ненужные строки условие проверки штрих – кода и номера паспорта.

**Задача 8.**

В главное окно 3-ей таблицей выводиться список лиц просрочившие заказ.

В таблице отображается результат отправки в СУБД такого запроса:

SELECT mydb.order.id\_order,TO\_DAYS(NOW())-TO\_DAYS(mydb.order.give\_date)-mydb.order.days as 'DifDays', \

readers.first\_name,readers.second\_name,readers.third\_name,readers.tel\_number,books.title FROM mydb.order,books, \

readers WHERE TO\_DAYS(NOW())-TO\_DAYS(mydb.order.give\_date) > mydb.order.days AND mydb.order.id\_book = books.id\_book \

AND mydb.order.id\_reader = readers.id\_reader AND mydb.order.receive\_date IS NULL ORDER BY mydb.order.give\_date DESC;

Выводится номер заказа, количество дней просрочки, имя, фамилия, отчество должника, его номер телефона, и название книги, которое ему надо вернуть. Вводится новая переменная которая рассчитывается на стороне БД и возвращает количество дней с момента просрочки. Выводом является результат отбора объединения таблиц order & books сусловиями на ненулевое число дней просрочки и равенство id\_reader и id\_biook и дата возврата не определена. Сортировка происходит по возрастанию количества дней просрочки.

**Задача 11.**

Разные настройки серверов БД фонда библиотеки, приводят к тому, что нужно реализовать подключение к ним.

При запуске программы в функции инициализации главного диалогового окна запускается модальное диалоговое окно (с вводом настроек для подключения к серверу БД). Пользователь не сможет работать с программой пока не аутентифицируется на сервере. Нажатие на кнопку отмена в этом окне приведёт к закрытию всей программы. Функция множественной проверки истинности настроек доступа к серверу.

void CBookingAccountingDlg::OnSetingsConnect()

{

Connect\_Settings\_Wnd dlg; //вызов окна для ввода настроек

INT\_PTR nRet = -1; //инициализация состояния окна

bool exit = false;

do {

nRet = dlg.DoModal(); //вызов модального окна ввода настроек

if (nRet == IDOK) { //запись значений в переменные настроек главного окна

addres\_database = dlg.m\_AddresServer\_Val;

name\_user = dlg.m\_NameUser\_Val;

password = dlg.m\_PasswordUser\_Val;

name\_database = dlg.m\_NameDB\_Val;

unix\_socket = dlg.m\_Socket\_Val;

port = atoi(dlg.m\_Port\_Val);

clientflag = atoi(dlg.m\_Flag\_Val);

}

if (nRet == IDCANCEL) { //выход из цикла, а в последствии из программы

exit = true;

break;

}

} while (!ConnectionSQL()); //проверка на подключение к БД

if (exit) OnBnClickedCancel(); //Программное нажатие на кнопку отмена главного окна

else {

updateTable("books");

Fill\_ctrl\_list(m\_ListBooks);

}

}

Программа имеет меню подключения в верхнем левом углу. При помощи него можно проверить не оборвана ли связь с сервером, или подключиться к другому фонду другой библиотеки.

**Установка программы**

Перед тем как запустить программу необходимо установить СУБД MySQL и построить БД по готовой схеме проектирования приложенной к проекту. Получить всю необходимую информацию для подключения к СУБД. Скомпилированная программа представляет из себя исполняемый .exe файл, готовый к запуску на платформе Windows.

**Результат**

Поставленные задачи в курсовой работе полностью выполнены: реализован графический интерфейсы программы, выбран СУБД, построена система баз данных, реализованы функции для: добавления книг в фонд, удаления книг из фонда, реализованы функции для создания и завершения заказа, для автоматизированного учёта должников, для поиска книг в фонде, для регистрации читателя в базе данных библиотеки. Реализованы ввод и проверка пользовательских данных (пароль, логин) для подключения к БД фонда.

Программа полностью соответствует своему функционалу, заявленному в задании на курсовую работу. Разработанная программа актуальна и имеет практическую значимость, т.к. облегчает труд работников библиотеки, обеспечивает  быстрый поиск, выдачу и возврат книг, а также ведет автоматизированный учет должников, поиск необходимой читателю литературы с использованием  электронных каталогов библиотеки.

В будущем можно добавить сохранение настроек, используемых в подключении к серверу в зашифрованном виде в файл, а также добавить базу операторов программы с индивидуальными логинами и паролями для сбора дополнительной статистики.

**Документация (руководство пользователя)**

При старте работы с программой необходимо ввести данные для подключения к БД. При отказе ввода данных программа завершается.

Главное окно приложения включает: список всех книг, список читателей взявших определенную книгу, список должников. При выделении интересующей нас книги в 1-ой таблице, во 2-ой отобразятся все читатели, взявшие эту книгу в аренду. Список должников обновляется при каждом входе в программу, в нём отображается количество дней просрочки, ФИО читателя, его номер телефона и название книги, которую он взял. Также в главном окне в левом нижнем углу отображается версия MySQL клиента, а верхнем левом углу экрана расположено меню подключения. В нём можно проверить связь с сервером БД или переключиться на другой.

При нажатии кнопки добавить/удалить книгу создаётся немодальное диалоговое окно с полями, необходимыми для заполнения, чтобы добавить книгу в фонд библиотеки. Если, какое-то поле будет не заполнено приложение создаст уведомление об этом. При успешном добавлении в БД книги также будет вызвано уведомление, сообщающие об успешном добавлении книги. Удаление книги можно произвести 2-мя способами либо удалить запись о книге полностью с последующим удалением всех заказов с ней или ввести конкретное число удаляемых книг. Для удаления записи о книге нужно выбрать соответствующий пункт, для удаления определённого числа надо вписать желаемое число в специальное поле. При удалении количества большего чем возможно удалить будет показано предупреждение об ошибке и операция не будет проведена. Для любого из видов удаления нужен только штрих – код книги.

Создать заказ. В данном диалоговом окне объединены 2 функции – создание заказа и добавление нового пользователя в БД библиотеки. В случае если пользователя нет в БД, то необходимо заполнить все его данные, дата регистрации берётся автоматически, по дате, которая установлена на компьютере в календаре, в случае не корректного ФИО (меньше 3 слов) или пустых полей программа выводит сообщение об ошибке и ввод надо повторить. Потом вводится количество дней аренды книги. Если поле подразумевает ввод только цифр то, в него нельзя ввести другие символы. Регистрация читателя в процессе осуществления заказа должна ускорить работу пользователя программы. Затем библиотекарь ищет нужную читателю книгу в фонде по названию, автору или году издания. В соответствующую таблицу выводятся все альтернативы. Поиск в таблице БД книг ведётся по подстрокам. Библиотекарь выделяет нужную книгу кликом мыши и нажимает кнопку *Совершить заказ.* Если читатель не был зарегистрирован, программа добавляет его посредством запроса к СУБД и выведет номер с которым он был добавлен (его можно расценивать как номер читательского билета), в случае если читатель уже в находиться в базе происходит проверка на наличие желаемой книги в фонде, и затем создание заказа в таблице БД и количество выбранной книги уменьшается на 1.В конце данной операции выводится окно, содержащие в себе номер только что добавленного заказа, для последующего использования библиотекарем(внесение в бланк заказа).

Завершение (Удаление) заказа. Завершить заказ можно, зная номер заказа или зная штрих-код книги и номер читательского билета клиента или номер его паспорта. Если читатель взял несколько одинаковых книг, то удаляется тот заказ, который был сделан раньше всех. Программа проверяет на пустоту необходимые комбинации полей и посылает запрос к БД. При корректном ответе от БД происходит вывод информации о успешном завершении заказа и количество возвращённой книги увеличивается на 1. Если книги по данному штрих – коду не было найдено будет выведено сообщение об этом.