МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

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

Факультет информационных технологий и компьютерной безопасности

Кафедра автоматизированных и вычислительных систем

КУРСОВОЙ ПРОЕКТ

по дисциплине «Объектно-ориентированное программирование»

Расчетно-пояснительная записка

Разработал студент _____ М. А. Романов
гр. бВМ-231 Подпись, дата Инициалы, фамилия

Руководитель _____ А. А. Акинин
Подпись, дата Инициалы, фамилия

Нормоконтролер _____ А. А. Акинина
Подпись, дата Инициалы, фамилия

Защищена _____ Оценка _____
Дата

Воронеж 2025

Задание

В рамках данной курсовой работы требуется разработать класс на языке C++ с консольным пользовательским интерфейсом. В классе должны быть реализованы конструктор по умолчанию, параметрический конструктор, конструктор копирования, деструктор и минимально необходимые методы set (…), get (…), show (…). На основе разработанного класса необходимо создать массив экземпляров класса с возможностью добавления, удаления и редактирования элементов этого массива. Редактирование должно осуществляться не всех свойств экземпляра класса принудительно, а любого свойства по выбору пользователя. Класс “Книга”. Свойства: автор, название, издательство, год выпуска, количество страниц.

В качестве пользовательского интерфейса кроме редактирования выбранных свойств объекта реализовать: вывод список книг заданного автора; вывод списка книг, с количеством страниц меньше заданного; список книг, выпущенных после заданного года.

В главе с реализацией будут приведены скриншоты с кодом приложения, но для удобства проверки весь код выложен на Github, его можно найти по ссылке:

https://github.com/Makarr-ivt/book_manager

1 Основы ООП на языке программирования C++

C++ является мультипарадигменным языком, но его центральным расширением по сравнению с C стали именно механизмы объектно-ориентированного программирования (ООП). ООП — это методология программирования, основанная на концепции «объектов», которые могут содержать данные (свойства, атрибуты) и код (методы, поведение).

В C++ реализация ООП базируется на четырех фундаментальных принципах:

– Инкапсуляция. Это механизм объединения данных и методов, работающих с этими данными, внутри единой структуры — класса. Инкапсуляция позволяет скрыть внутреннюю реализацию объекта от внешнего мира, предоставляя контролируемый интерфейс взаимодействия. В C++ это достигается с помощью модификаторов доступа (public, private, protected).

– Наследование. Это механизм, позволяющий создавать новые классы (классы-наследники или производные классы) на основе существующих классов (базовые классы). Наследники автоматически получают свойства и методы базового класса и могут добавлять свои собственные или переопределять существующие. Это способствует повторному использованию кода.

– Полиморфизм. Этот принцип предлагает использовать объекты разных классов, имеющих общее базовое определение, единообразным образом. В C++ полиморфизм в основном реализуется через виртуальные функции и указатели/ссылки на базовый класс, что позволяет определить, какая конкретная реализация метода будет вызвана во время выполнения программы.

– Абстракция. Это процесс выделения значимых характеристик объекта и отбрасывания несущественных деталей. В C++ абстракция реализуется через использование классов, абстрактных базовых классов (с чисто виртуальными функциями) и интерфейсов, что позволяет работать с сущностями на более высоком концептуальном уровне.

2 Реализация приложения

В нашем приложении требуется создать только класс “Книга”. Класс должен включать в себя следующие свойства:

- author — ФИО автора книги;
- title — название книги;
- publisher — имя издательства, напечатавшего книгу;
- year — год издания;
- pages — количество страниц в книге.

Первые три свойства сделаем типа std::string, а год издания и количество страниц целочисленным типом int. В соответствии с принципом инкапсуляции, все свойства находятся под модификатором private. Получить или изменить их значения можно будет с помощью соответствующих методов: геттеров и сеттеров. Так для каждого свойства мы создаем 2 метода подобного вида:

```
void setAuthor(const string& a);

string getAuthor() const;
```

Заметим, что все геттеры имеют модификатор const. Он обозначает, что данные методы не изменяют состояние объекта. Для свойств строкового типа в сеттерах аргумент имеет тип const string&, то есть константную ссылку на строку вместо простого string. Это сделано для эффективного использования памяти программы, избежания чрезмерного копирования строк.

Кроме сеттеров и геттеров нам потребуется константный метод void show() const для вывода информации об объекте. Также не забываем про конструкторы и деструктор. Конструктор — это метод, который вызывается при создании объекта. Он передает необходимые данные свойствам объекта, а также может выполнять дополнительную логику, если потребуется. Конструкторы бывают разных типов:

– Конструктор по умолчанию. Это конструктор без аргументов, который позволяет создать объект с предустановленными значениями свойств.

– Параметрический конструктор. Позволяет при создании объекта указать конкретные значения свойств. Можно перегрузить этот конструктор, чтобы можно было задать объект по разному набору параметров (например, указать только имя автора, или только автора и название, а остальные поля заполнить значениями по умолчанию).

– Конструктор копирования. Принимает в качестве аргумента ссылку на другой объект того же класса и создает объект, повторяющий все свойства исходного.

Деструктор – это метод, который вызывается при удалении объекта из памяти программы. Часто используется деструктор по умолчанию, который сам эффективно очищает память. Но также деструктор позволяет реализовать дополнительную логику.

Для создания класса “Книга” создадим отдельный заголовочный файл Book.h, в котором объявим свойства и методы класса. В файле Book.cpp реализуем все методы. Код заголовочного файла на рисунке 1, код Book.cpp на рисунке 2, рассмотрим его подробнее. Видим, что логика всех методов достаточно проста и почти везде умещается в одну строку. На 3 строке кода конструктор по умолчанию, на 5 параметрический конструктор, на 8 конструктор копирования, деструктор на 12 строке обозначается тильдой (~) и не имеет реализации. Это значит, что в программе используется деструктор по умолчанию. На строках 14 –18 сеттеры, на 20 – 24 геттеры. В конце реализован метод show.

```cpp
#pragma once
#include <string>
#include <iostream>

using namespace std;

class Book {
private:
    string author;
    string title;
    string publisher;
    int year;
    int pages;

public:
    Book();
    Book(const string& a, const string& t, const string& p, int y, int pg);
    Book(const Book& other);

    ~Book();

    void setAuthor(const string& a);
    void setTitle(const string& t);
    void setPublisher(const string& p);
    void setYear(int y);
    void setPages(int pg);

    string getAuthor() const;
    string getTitle() const;
    string getPublisher() const;
    int getYear() const;
    int getPages() const;

    void show() const;
};

```

Рисунок 1 – код заголовочного файла Book.h

```cpp
#include "Book.h"

Book::Book() : author(""), title(""), publisher(""), year(0), pages(0) {}

Book::Book(const string& a, const string& t, const string& p, int y, int pg)
    : author(a), title(t), publisher(p), year(y), pages(pg) {}

Book::Book(const Book& other)
    : author(other.author), title(other.title),
      publisher(other.publisher), year(other.year), pages(other.pages) {}

Book::~Book() {}

void Book::setAuthor(const string& a) { author = a; }
void Book::setTitle(const string& t) { title = t; }
void Book::setPublisher(const string& p) { publisher = p; }
void Book::setYear(int y) { year = y; }
void Book::setPages(int pg) { pages = pg; }

string Book::getAuthor() const { return author; }
string Book::getTitle() const { return title; }
string Book::getPublisher() const { return publisher; }
int Book::getYear() const { return year; }
int Book::getPages() const { return pages; }

void Book::show() const {
    cout << "Автор: " << author << "\n"
         << "Название: " << title << "\n"
         << "Издательство: " << publisher << "\n"
         << "Год: " << year << "\n"
         << "Страниц: " << pages << "\n"
         << "------------------------\n";
}

```

Рисунок 2 – код исполняемого файла Book.cpp

Мы получили класс, который написан в двух файлах на 34 и 36 строки с учетом пустых. Он содержит всю необходимую логику, чтобы использовать его в программе. Класс простой и хорошо читается, его можно будет легко редактировать или расширять в будущем. Теперь приступим к реализации пользовательского интерфейса, где будем использовать объекты нашего класса.

Нам требуется создать массив объектов класса Book и выполнять над массивом действия. Можно было бы создавать этот массив сразу в главной функции main(), и дописать несколько функций, которые принимали бы ссылку на этот массив и изменяли его. Или сделать массив глобальным объектом, доступным для изменения в любой части программы. Однако более грамотным решением будет создать новый класс BookManager. У него будет одно свойство: массив объектов класса Book. Также этот класс будет содержать все методы для взаимодействия с объектами этого массива. Такой подход позволит держать весь интерфейс в одном месте, что удобно для возможного будущего расширения приложения.

В классе BookManager мы реализуем методы для базовых действий с книгами: создание, удаление, редактирование выбранного свойства, вывод всех книг в массиве. Кроме того, в соответствии с заданием реализуем методы, которые позволят выводить книги по имени автора, по году издания и по количеству страниц. Главным методом класса будет метод run(), содержащий основное меню приложения с выбором действия.

Кроме того, в процессе разработки было принято решение выделить 3 утилитных метода: clearInput() для очистки буфера ввода, pauseAndClear() для ожидания действия пользователя и последующей очистки терминала, inputInt() для ввода целого числа с проверкой на корректность и ограничениями. Правильным решением было бы выделить эти методы в третий утилитный класс, потому что

потенциально они могли бы использоваться в других местах приложения при расширении. Однако в рамках данной курсовой работы оставим эти методы внутри класса BookManager, при необходимости перенести их в новый класс не составит труда. Все методы класса кроме run() сделаны приватными, так как вызываются внутри класса. Мы не объявляем конструкторы и деструкторы, при компляции будут созданы констуктор и деструктор по умолчанию, которых в нашем случае достаточно. Класс объявлен в заголовочном файле BookManager.h, код на рисунке 3.

```
#pragma once
#include "Book.h"
#include <vector>
#include <climits>  // Для INT_MIN и INT_MAX

class BookManager {
private:
    vector<Book> books;
    void clearInput() const;
    void pauseAndClear() const;
    int inputInt(const string& prompt, int min = INT_MIN, int max = INT_MAX) const;

    void addBook();
    void deleteBook();
    void editBook();
    void showAllBooks() const;

    void showBooksByAuthor() const;
    void showBooksByPages() const;
    void showBooksAfterYear() const;
public:

    void run();
};

```

Рисунок 3 – Код файла BookManager.h, объявление класса

Реализация методов не содержит особенно сложной логики или каких-либо продвинутых функций C++. Для добавления и удаления объектов массива используются стандартные методы вектора emplace_back() и erase(). с помощью базовых методов cin.clear() и cin.ignore(), для очистки терминала используем

system(“cls”), так как приложение рассчитано на терминал Shell. Ввод числовых значений происходит через cin >>, ввод строковых значений — через getline(). Доступ к свойствам объектов класса Book происходит через геттеры и сеттеры, которые мы реализовали ранее. Вывод информации о каждой книге также реализован нами в классе Book. В остальном при реализации методов используются стандартные конструкции языка: условные операторы, циклы, switch-case и т.д. Все методы реализованы в файле BookManager.cpp, код представлен на рисунках 4 – 11.

```cpp
#include "BookManager.h"
#include <iostream>
#include <limits> // для numeric_limits<streamsize>::max()
#include <cstdlib>  // для system("cls")
#include <string>

using namespace std;

void BookManager::clearInput() const {
    cin.clear();
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
}

// Пауза с ожиданием нажатия клавиши и очистка экрана
void BookManager::pauseAndClear() const {
    cout << "\nНажмите Enter для продолжения...";
    clearInput();
    system("cls");
}

// ввод целого числа с проверкой
int BookManager::inputInt(const string& prompt, int min, int max) const {
    int value;
    bool isValid = false;

    do {
        cout << prompt;
        if (cin >> value) {
            if (value >= min && value <= max) {
                isValid = true;
            } else {
                cout << "Ошибка: число должно быть в диапазоне от "
                     << min << " до " << max << "!\n";
            }
        } else {
            cout << "Ошибка: введите корректное целое число!\n";
        }
        clearInput();
    } while (!isValid);

    return value;
}

```

Рисунок 4 – файл BookManager.cpp, реализация методов clearInput(), pauseAndClear(), InputInt()

```
43
44  void BookManager::addBook() {
45      string author, title, publisher;
46      int year, pages;
47
48      cout << "Введите автора: ";
49      getline(cin, author);
50
51      cout << "Введите название: ";
52      getline(cin, title);
53
54      cout << "Введите издательство: ";
55      getline(cin, publisher);
56
57      year = inputInt("Введите год издания: ", 0, 3000);
58
59      pages = inputInt("Введите количество страниц: ", 1, 10000);
60
61      books.emplace_back(author, title, publisher, year, pages);
62      cout << "\nКнига добавлена успешно!\n";
63  }
64
65  void BookManager::deleteBook() {
66      if (books.empty()) {
67          cout << "Список книг пуст!\n";
68          pauseAndClear();
69          return;
70      }
71
72      showAllBooks();
73      int index = inputInt("\nВведите номер книги для удаления (1-" +
74                           to_string(books.size()) + "): ",
75                           1, books.size());
76
77      books.erase(books.begin() + index - 1);
78      cout << "Книга удалена успешно!\n";
79  }
80
```

Рисунок 5 – Реализация методов addBook() и deleteBook() класса BookMAnager

```
80
81  void BookManager::editBook() {
82      if (books.empty()) {
83          cout << "Список книг пуст!\n";
84          pauseAndClear();
85          return;
86      }
87
88      showAllBooks();
89      int index = inputInt("\nВведите номер книги для редактирования (1-" +
90                           to_string(books.size()) + "): ",
91                           1, books.size());
92
93      Book& book = books[index - 1];
94      int choice;
95
96      do {
97          cout << "\n=== РЕДАКТИРОВАНИЕ КНИГИ ===\n";
98          cout << "Текущая информация о книге:\n";
99          book.show();
100
101         cout << "\nЧто вы хотите изменить?\n"
102                  << "1. Автора\n"
103                  << "2. Название\n"
104                  << "3. Издательство\n"
105                  << "4. Год издания\n"
106                  << "5. Количество страниц\n"
107                  << "6. Вернуться в меню\n"
108                  << "Выберите опцию: ";
109         cin >> choice;
110         clearInput();
111
112         string strValue;
113         int intValue;
114
115         switch (choice) {
116         case 1:
```

Рисунок 6 – Первая часть метода BookManager.editBook()

```
        string strValue;
        int intValue;

        switch (choice) {
        case 1:
            cout << "Текущий автор: " << book.getAuthor() << "\n";
            cout << "Введите нового автора: ";
            getline(cin, strValue);
            book.setAuthor(strValue);
            cout << "Автор изменен успешно!\n";
            break;
        case 2:
            cout << "Текущее название: " << book.getTitle() << "\n";
            cout << "Введите новое название: ";
            getline(cin, strValue);
            book.setTitle(strValue);
            cout << "Название изменено успешно!\n";
            break;
        case 3:
            cout << "Текущее издательство: " << book.getPublisher() << "\n";
            cout << "Введите новое издательство: ";
            getline(cin, strValue);
            book.setPublisher(strValue);
            cout << "Издательство изменено успешно!\n";
            break;
        case 4:
            cout << "Текущий год издания: " << book.getYear() << "\n";
            intValue = inputInt("Введите новый год издания: ", 0, 3000);
            book.setYear(intValue);
            cout << "Год изменен успешно!\n";
            break;
        case 5:
            cout << "Текущее количество страниц: " << book.getPages() << "\n";
            intValue = inputInt("Введите новое количество страниц: ", 1, 10000);
            cout << "Количество страниц изменено успешно!\n";
            break;
        case 6:
            return;
        default:
            cout << "Неверный выбор! Попробуйте снова.\n";
        }
        pauseAndClear();
    } while (choice != 6);
}

```

Рисунок 7 – Вторая часть метода BookManager.editBook(), цикл и switch-case

```
156
157 void BookManager::showAllBooks() const {
158     if (books.empty()) {
159         cout << "Список книг пуст!\n";
160         return;
161     }
162
163     cout << "\n=== СПИСОК ВСЕХ КНИГ ===\n";
164     cout << "Всего книг: " << books.size() << "\n\n";
165     for (size_t i = 0; i < books.size(); ++i) {
166         cout << "Книга #" << i + 1 << ":\n";
167         books[i].show();
168     }
169 }
170
171 void BookManager::showBooksByAuthor() const {
172     if (books.empty()) {
173         cout << "Список книг пуст!\n";
174         return;
175     }
176
177     string author;
178     cout << "Введите автора для поиска: ";
179     getline(cin, author);
180
181     cout << "\n=== КНИГИ АВТОРА: " << author << " ===\n";
182     int count = 0;
183
184     for (const auto& book : books) {
185         if (book.getAuthor() == author.) {
186             book.show();
187             count++;
188         }
189     }
190
191     if (!count) {
192         cout << "Книги данного автора не найдены.\n";
193     } else {
194         cout << "Найдено книг: " << count << "\n";
195     }
196 }
197
```

Рисунок 8 – Реализация методов BookManager.showAllBooks() и BookManager.showBooksByAuthor()

```
198 void BookManager::showBooksByPages() const {
199     if (books.empty()) {
200         cout << "Список книг пуст!\n";
201         return;
202     }
203
204     int maxPages = inputInt("Введите максимальное количество страниц: ", 1, 10000);
205
206     if (maxPages <= 0) {
207         cout << "Неверное количество страниц!\n";
208         return;
209     }
210
211     cout << "\n=== КНИГИ С КОЛИЧЕСТВОМ СТРАНИЦ МЕНЕЕ " << maxPages << " ===\n";
212     int count = 0;
213
214     for (const auto& book : books) {
215         if (book.getPages() < maxPages) {
216             book.show();
217             count++;
218         }
219     }
220
221     if (!count) {
222         cout << "Книги с указанным количеством страниц не найдены.\n";
223     } else {
224         cout << "Найдено книг: " << count << "\n";
225     }
226 }
227
```

Рисунок 9 – Реализация метода BookManager.showBooksByPages()

```
227
228 void BookManager::showBooksAfterYear() const {
229     if (books.empty()) {
230         cout << "Список книг пуст!\n";
231         return;
232     }
233
234     int minYear;
235     cout << "Введите минимальный год издания: ";
236     cin >> minYear;
237     clearInput();
238
239     cout << "\n=== КНИГИ, ИЗДАННЫЕ ПОСЛЕ " << minYear << " ГОДА ===\n";
240     int count = 0;
241
242     for (const auto& book : books) {
243         if (book.getYear() > minYear) {
244             book.show();
245             count++;
246         }
247     }
248
249     if (!count) {
250         cout << "Книги, изданные после указанного года, не найдены.\n";
251     } else {
252         cout << "Найдено книг: " << count << "\n";
253     }
254 }
255
```

Рисунок 10 – Реализация метода BookManager.showBooksAfterYear()

```
255
256 void BookManager::run() {
257     int choice;
258
259     do {
260         cout << "\n=== МЕНЕДЖЕР КНИГ ===\n"
261                  << "1. Добавить книгу\n"
262                  << "2. Удалить книгу\n"
263                  << "3. Редактировать книгу\n"
264                  << "4. Показать все книги\n"
265                  << "5. Найти книги по автору\n"
266                  << "6. Найти книги по количеству страниц\n"
267                  << "7. Найти книги по году издания\n"
268                  << "8. Выход\n"
269                  << "========================\n"
270                  << "Выберите опцию (1-8): ";
271
272         choice = inputInt("Выберите опцию (1-8): ", 1, 8);
273
274         switch (choice) {
275         case 1:
276             addBook();
277             pauseAndClear();
278             break;
279         case 2:
280             deleteBook();
281             pauseAndClear();
282             break;
283         case 3:
284             editBook();
285             pauseAndClear();
286             break;
287         case 4:
288             showAllBooks();
289             pauseAndClear();
290             break;
291         case 5:
292             showBooksByAuthor();
293             pauseAndClear();
294             break;
295         case 6:
296             showBooksByPages();
297             pauseAndClear();
298             break;
299         case 7:
300             showBooksAfterYear();
301             pauseAndClear();
302             break;
303         case 8:
304             cout << "\nВыход из программы...\n";
305             break;
306         }
307     } while (choice != 8);
308 }
309
```

Рисунок 11 – Конец файла BookManager.cpp, метод run()

Последняя и самая простая часть приложения — функция main(), которая не содержит никакой логики и является только точкой входа в программу. В ней создается объект класса BookManager и запускается метод run(), больше ничего. Код файла main.cpp представлен на рисунке 12.

```
1   #include "BookManager.h"
2
3   int main() {
4       system("chcp 65001"); // русский язык для терминала
5
6       BookManager manager;
7       manager.run();
8
9       return 0;
10  }
11
```

Рисунок 12 – Файл main.cpp

3 Результаты работы приложения

Программа готова. Примеры работы представлены на рисунках 13 – 20. Кроме скриншотов, в репозитории Github находится .zip архив с записью экрана на 3 минуты, которая демонстрирует работу приложения.

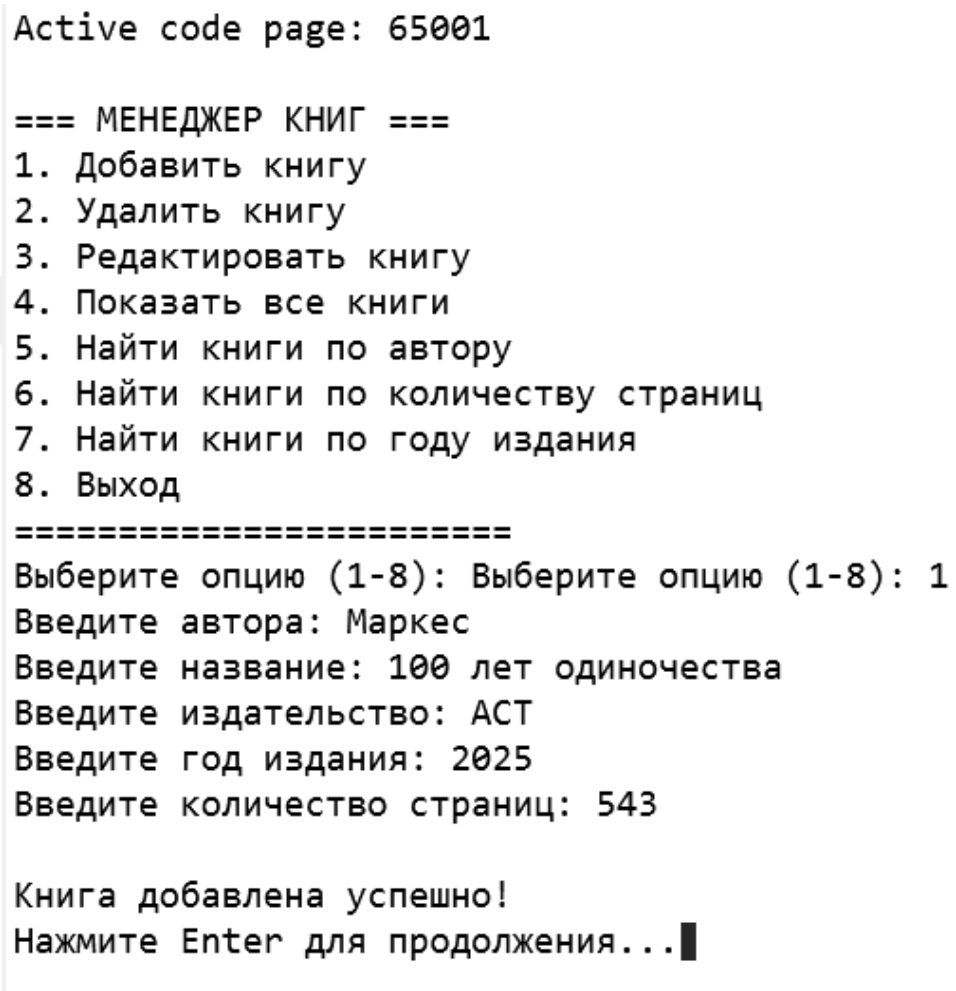

Рисунок 13 – Работа программы, создание книги

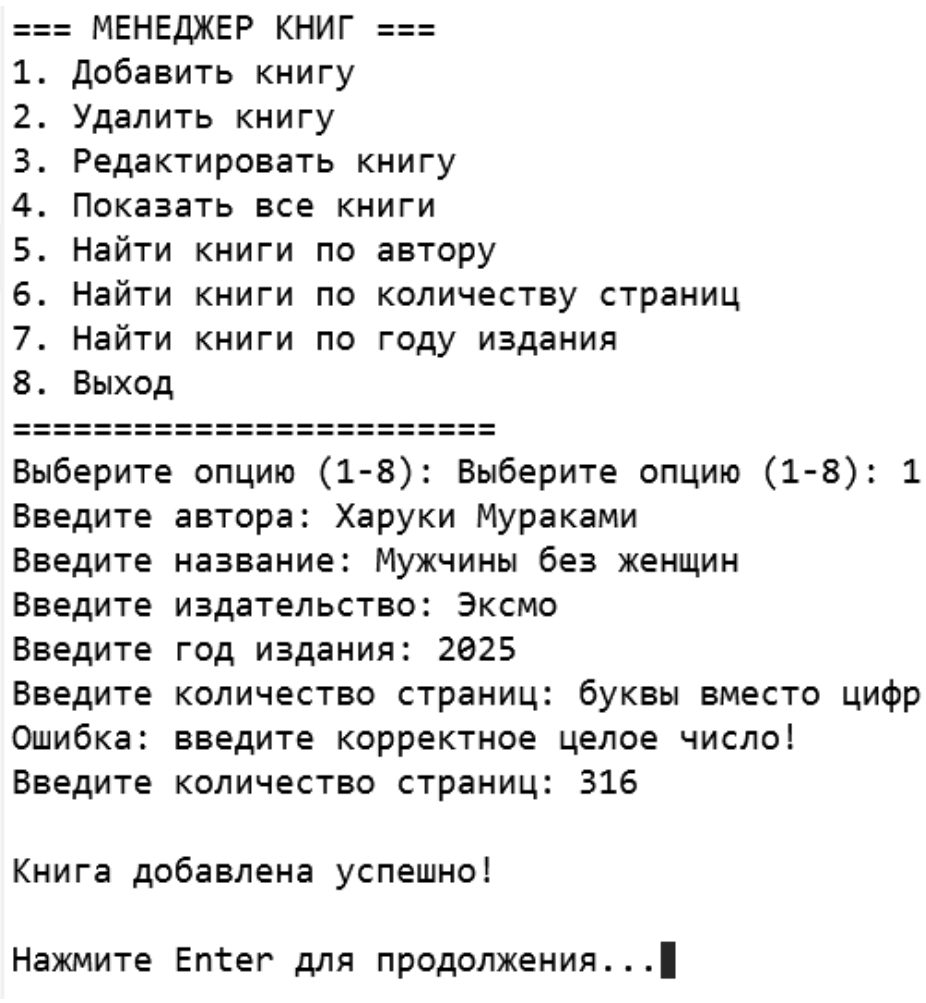

Рисунок 14 – Работа программы, создание книги с проверкой обработки ввода

```
Страниц: 316
------------------------
Книга #3:
Автор: Жан Поль Сартр
Название: Мухи
Издательство: АСТ
Год: 2023
Страниц: 93
------------------------
Книга #4:
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: АСТ
Год: 2023
Страниц: 193
------------------------

Введите номер книги для редактирования (1-4): 4

=== РЕДАКТИРОВАНИЕ КНИГИ ===
Текущая информация о книге:
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: АСТ
Год: 2023
Страниц: 193
------------------------

Что вы хотите изменить?
1. Автора
2. Название
3. Издательство
4. Год издания
5. Количество страниц
6. Вернуться в меню
Выберите опцию: 3
Текущее издательство: АСТ
Введите новое издательство: Эксмо
Издательство изменено успешно!

Нажмите Enter для продолжения...
```

Рисунок 15 – Работа программы, редактирование издательства

```
Книга #1:
Автор: Маркес
Название: 100 лет одиночества
Издательство: АСТ
Год: 2025
Страниц: 543
------------------------
Книга #2:
Автор: Харуки Мураками
Название: Мужчины без женщин
Издательство: Эксмо
Год: 2025
Страниц: 316
------------------------
Книга #3:
Автор: Жан Поль Сартр
Название: Мухи
Издательство: АСТ
Год: 2023
Страниц: 93
------------------------
Книга #4:
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: Эксмо
Год: 2023
Страниц: 193
------------------------
Книга #5:
Автор: Артур Шопенгауэр
Название: Мир как воля и представление том 2
Издательство: Академический проект
Год: 2019
Страниц: 567
------------------------

Введите номер книги для удаления (1-5): 1
Книга удалена успешно!

Нажмите Enter для продолжения...
```

Рисунок 16 – Работа программы, удаление книги

```
4. Показать все книги
5. Найти книги по автору
6. Найти книги по количеству страниц
7. Найти книги по году издания
8. Выход
========================
Выберите опцию (1-8): Выберите опцию (1-8): 4

=== СПИСОК ВСЕХ КНИГ ===
Всего книг: 4

Книга #1:
Автор: Харуки Мураками
Название: Мужчины без женщин
Издательство: Эксмо
Год: 2025
Страниц: 316
------------------------
Книга #2:
Автор: Жан Поль Сартр
Название: Мухи
Издательство: АСТ
Год: 2023
Страниц: 93
------------------------
Книга #3:
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: Эксмо
Год: 2023
Страниц: 193
------------------------
Книга #4:
Автор: Артур Шопенгауэр
Название: Мир как воля и представление том 2
Издательство: Академический проект
Год: 2019
Страниц: 567
------------------------

Нажмите Enter для продолжения...
```

Рисунок 17 – Работа программы, вывод списка книг без удаленной книги

```
=== МЕНЕДЖЕР КНИГ ===
1. Добавить книгу
2. Удалить книгу
3. Редактировать книгу
4. Показать все книги
5. Найти книги по автору
6. Найти книги по количеству страниц
7. Найти книги по году издания
8. Выход
========================
Выберите опцию (1-8): Выберите опцию (1-8): 5
Введите автора для поиска: Жан Поль Сартр

=== КНИГИ АВТОРА: Жан Поль Сартр ===
Автор: Жан Поль Сартр
Название: Мухи
Издательство: АСТ
Год: 2023
Страниц: 93
------------------------
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: Эксмо
Год: 2023
Страниц: 193
------------------------
Найдено книг: 2

Нажмите Enter для продолжения...
```

Рисунок 18 – Работа программы, поиск по автору

```
=== МЕНЕДЖЕР КНИГ ===
1. Добавить книгу
2. Удалить книгу
3. Редактировать книгу
4. Показать все книги
5. Найти книги по автору
6. Найти книги по количеству страниц
7. Найти книги по году издания
8. Выход
========================
Выберите опцию (1-8): Выберите опцию (1-8): 6
Введите максимальное количество страниц: 350

=== КНИГИ С КОЛИЧЕСТВОМ СТРАНИЦ МЕНЕЕ 350 ===
Автор: Харуки Мураками
Название: Мужчины без женщин
Издательство: Эксмо
Год: 2025
Страниц: 316
------------------------
Автор: Жан Поль Сартр
Название: Мухи
Издательство: АСТ
Год: 2023
Страниц: 93
------------------------
Автор: Жан Поль Сартр
Название: Затворники Альтоны
Издательство: Эксмо
Год: 2023
Страниц: 193
------------------------
Найдено книг: 3

Нажмите Enter для продолжения...
```

Рисунок 19 – Работа программы, поиск по количеству страниц

```
=== МЕНЕДЖЕР КНИГ ===
1. Добавить книгу
2. Удалить книгу
3. Редактировать книгу
4. Показать все книги
5. Найти книги по автору
6. Найти книги по количеству страниц
7. Найти книги по году издания
8. Выход
========================
Выберите опцию (1-8): Выберите опцию (1-8): 7
Введите минимальный год издания: 2024

=== КНИГИ, ИЗДАННЫЕ ПОСЛЕ 2024 ГОДА ===
Автор: Харуки Мураками
Название: Мужчины без женщин
Издательство: Эксмо
Год: 2025
Страниц: 316
------------------------
Найдено книг: 1

Нажмите Enter для продолжения...
```

Рисунок 20 – Работа программы, поиск по году издания

Заключение

В результате выполнения задания были разработаны 2 класса для работы с книгами. Библиотека позволяет создавать, редактировать и удалять экземпляры книги с информацией об авторе, названии, издательстве, количестве страниц. Реализована возможность вывода информации о книгах, поиск по имени автора, количеству страниц, году издательства. Приложение спроектировано так, чтобы его было просто поддерживать и расширять.

Выполнение задания для ознакомительной практики позволило продемонстрировать навыки владения языком C++, знания парадигмы ООП, умения создавать рабочие консольные приложения.