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, искусственный интеллект и системы управления» (ИУ)
КАФЕДРА «Информационная безопасность» (ИУ8)

Домашнее задание №3

ПО КУРСУ

«Алгоритмические языки»

Студент ИУ8-21 (Группа) Г. А. Карев (И. О. Фамилия)

Преподаватель: В. В. Соборова (И.О. Фамилия)

Задание:

Разработать программу, работающую с объектами классов. Используя интерфейс командной строки, реализовать следующие режимы работы: «Ввод нового объекта и добавление его в контейнер», «Поиск объекта в контейнере по значениям полей с печатью данных о найденных объектах», «Редактирование объекта», «Удаление объекта из контейнера», «Сохранение данных всех объектов в файле», «Чтение данных объектов из файла», «Сортировка объектов контейнера по выбранному полю для list», «Печать списка объектов». Предусмотреть обработку исключений, возможные исключения определить самостоятельно. В функции main должен быть главный поток, который создает консольное меню для выбора режима, режимы, требующие взаимодействия с пользователем (ввод нового объекта, редактирование объекта, поиск и печатью, печать списка объектов и т.п.) выполняются в этом главном потоке. Режимы, не требующие взаимодействия с пользователем (удаление, сохранение в файле и чтение из файла), выполняются в отдельном потоке, созданном в главном, при этом обеспечить синхронизацию при доступе к данным объектов.

Использование обработки исключений и потоковой многозадачности обязательны для отличной оценки.

Тема задания	Контейнер		
	Список list	Неупорядоченное множество unordered_set	Упорядоченное множество set
	Номера вариантов		
Класс домов и класс улиц для учета поступлений квартплаты.	1	8	15
Класс студентов и класс учебных групп для учета успеваемости по итогам одного семестра, предусмотреть расчет среднего балла для группы по всем дисциплинам и по отдельной дисциплине, печать отличников и задолженников.	2	9	16
Класс сотрудников и класс структурных подразделений (отделов и др.) для отдела кадров организации.	3	10	17
Класс сотрудников и класс структурных подразделений (отделов и др.) для учета начислений зарплаты в бухгалтерии.	4	11	18
Класс книга и класс библиотека для учета книг в библиотеке.	5	12	18
Класс файлов и класс каталогов файлов, предусмотреть поиск по имени файла. Предусмотреть операции перемещения файлов, их добавления и удаления, поиска, переименования, сравнения и объединения каталогов и т.д.	6	13	20
Класс домов и класс улиц для получения списка избирателей для участия в выборах.	7	14	21

Код:

Library.h

```
#pragma once
#include "Book.h"
#include <unordered_set>
#include <mutex>
#include <string>

class Library {
    std::unordered_set<Book> books;
    mutable std::mutex mtx;

public:
    void addBook(const Book&);
    bool removeBook(const std::string& title);
    Book* findBook(const std::string& title);
    void editBook(const std::string& title);
    void listBooks() const;
    void sortByTitle() const;
    void sortByAuthor() const;
    void sortByGenre() const;
    void saveToFile(const std::string& filename) const;
    void loadFromFile(const std::string& filename);
};
```

Library.cpp

```
#include "Library.h"
#include <fstream>
#include <iostream>
#include <vector>
#include <algorithm>

void Library::addBook(const Book& b) {
    std::lock_guard<std::mutex> lock(mtx);
    books.insert(b);
}

bool Library::removeBook(const std::string& title) {
    std::lock_guard<std::mutex> lock(mtx);
    for (auto it = books.begin(); it != books.end(); ++it) {
        if (it->getTitle() == title) {
            books.erase(it);
            return true;
        }
    }
    return false;
}

Book* Library::findBook(const std::string& title) {
    std::lock_guard<std::mutex> lock(mtx);
    for (auto& b : books) {
        if (b.getTitle() == title)
            return const_cast<Book*>(&b);
    }
    return nullptr;
}

void Library::editBook(const std::string& title) {
    Book* book = findBook(title);
    if (!book) {
        std::cout << "Книга не найдена!\n";
        return;
    }

    std::string newTitle, newAuthor;
    int newYear, genre;
    std::cin.ignore();
    std::cout << "Новое название: ";
    std::getline(std::cin, newTitle);
    std::cout << "Новый автор: ";
    std::getline(std::cin, newAuthor);
    std::cout << "Новый год: ";
    std::cin >> newYear;
    std::cout << "Новый жанр:\n"
              << " 0 – Художественная литература\n"
              << " 1 – Документальная литература\n"
              << " 2 – Научная литература\n"
              << " 3 – Биография\n"
              << "Введите номер жанра: ";
    std::cin >> genre;

    book->setTitle(newTitle);
    book->setAuthor(newAuthor);
    book->setYear(newYear);
    book->setGenre(Book::genreFromInt(genre));
}

void Library::listBooks() const {
    std::lock_guard<std::mutex> lock(mtx);
    std::cout << "\n  Название               |       Автор        |  Год  |   Жанр
\n"
              << "-----------------------|--------------------|-------|---------
\n";
```

```
    for (const auto& b : books)
        std::cout << b << std::endl;
}

void Library::sortByTitle() const {
    std::lock_guard<std::mutex> lock(mtx);
    std::vector<Book> sorted(books.begin(), books.end());
    std::sort(sorted.begin(), sorted.end(), [](const Book& a, const Book& b) {
        return a.getTitle() < b.getTitle();
    });

    std::cout << "\n  Название               |       Автор        |  Год  |   Жанр
\n"
              << "-------------------------|--------------------|-------|----------
\n";
    for (const auto& b : sorted)
        std::cout << b << std::endl;
}

void Library::sortByAuthor() const {
    std::lock_guard<std::mutex> lock(mtx);
    std::vector<Book> sorted(books.begin(), books.end());
    std::sort(sorted.begin(), sorted.end(), [](const Book& a, const Book& b) {
        return a.getAuthor() < b.getAuthor();
    });

    std::cout << "\n  Название               |       Автор        |  Год  |   Жанр
\n"
              << "-------------------------|--------------------|-------|----------
\n";
    for (const auto& b : sorted)
        std::cout << b << std::endl;
}

void Library::sortByGenre() const {
    std::lock_guard<std::mutex> lock(mtx);
    std::vector<Book> sorted(books.begin(), books.end());
    std::sort(sorted.begin(), sorted.end(), [](const Book& a, const Book& b) {
        return a.getGenre() < b.getGenre();
    });

    std::cout << "\n  Название               |       Автор        |  Год  |   Жанр
\n"
              << "-------------------------|--------------------|-------|----------
\n";
    for (const auto& b : sorted)
        std::cout << b << std::endl;
}

void Library::saveToFile(const std::string& filename) const {
    std::lock_guard<std::mutex> lock(mtx);
    std::ofstream out(filename);
    for (const auto& b : books)
        out << b.getTitle() << ";" << b.getAuthor() << ";" << b.getYear() << ";" <<
static_cast<int>(b.getGenre()) << "\n";
}

void Library::loadFromFile(const std::string& filename) {
    std::lock_guard<std::mutex> lock(mtx);
    std::ifstream in(filename);
    if (!in) throw std::runtime_error("Файл не найден!");

    std::string title, author;
    int year, genre;
    while (std::getline(in, title, ';') &&
           std::getline(in, author, ';') &&
           (in >> year).ignore() &&
           (in >> genre).ignore()) {
        books.emplace(title, author, year, Book::genreFromInt(genre));
    }
```

```
}
```

Book.h

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

enum class BookGenre { Fiction = 0, NonFiction, Science, Biography, Unknown };

class Book {
    std::string title;
    std::string author;
    int year;
    BookGenre genre;

public:
    Book();
    Book(const std::string&, const std::string&, int, BookGenre);

    bool operator==(const Book& other) const;

    std::string getTitle() const;
    std::string getAuthor() const;
    int getYear() const;
    BookGenre getGenre() const;

    void setTitle(const std::string&);
    void setAuthor(const std::string&);
    void setYear(int);
    void setGenre(BookGenre);

    std::string genreToString() const;
    static BookGenre genreFromInt(int g);

    friend std::ostream& operator<<(std::ostream&, const Book&);
};

namespace std {
    template<>
    struct hash<Book> {
        size_t operator()(const Book& b) const {
            return hash<string>()(b.getTitle() + b.getAuthor());
        }
    };
}
```

Book.cpp

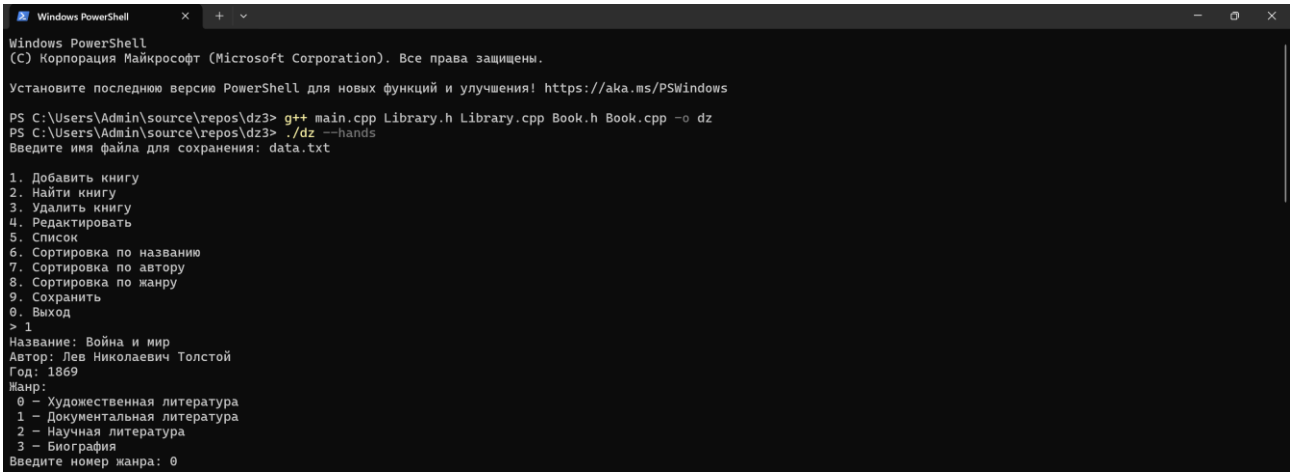
```cpp
#include "Book.h"

Book::Book() : title(""), author(""), year(0), genre(BookGenre::Unknown) {}

Book::Book(const std::string& t, const std::string& a, int y, BookGenre g)
    : title(t), author(a), year(y), genre(g) {}

bool Book::operator==(const Book& other) const {
    return title == other.title && author == other.author;
}

std::string Book::getTitle() const { return title; }
std::string Book::getAuthor() const { return author; }
int Book::getYear() const { return year; }
BookGenre Book::getGenre() const { return genre; }

void Book::setTitle(const std::string& t) { title = t; }
void Book::setAuthor(const std::string& a) { author = a; }
void Book::setYear(int y) { year = y; }
void Book::setGenre(BookGenre g) { genre = g; }

std::string Book::genreToString() const {
    switch (genre) {
        case BookGenre::Fiction: return "Худ. лит.";
        case BookGenre::NonFiction: return "Док. лит.";
        case BookGenre::Science: return "Науч. лит.";
        case BookGenre::Biography: return "Биография";
        default: return "Неизв.";
    }
}

BookGenre Book::genreFromInt(int g) {
    switch (g) {
        case 0: return BookGenre::Fiction;
        case 1: return BookGenre::NonFiction;
        case 2: return BookGenre::Science;
        case 3: return BookGenre::Biography;
        default: return BookGenre::Unknown;
    }
}

std::ostream& operator<<(std::ostream& os, const Book& b) {
    os.width(25); os << std::left << b.title << "| ";
    os.width(20); os << std::left << b.author << "| ";
    os.width(5);  os << std::left << b.year << "| ";
    os << b.genreToString();
    return os;
}
```

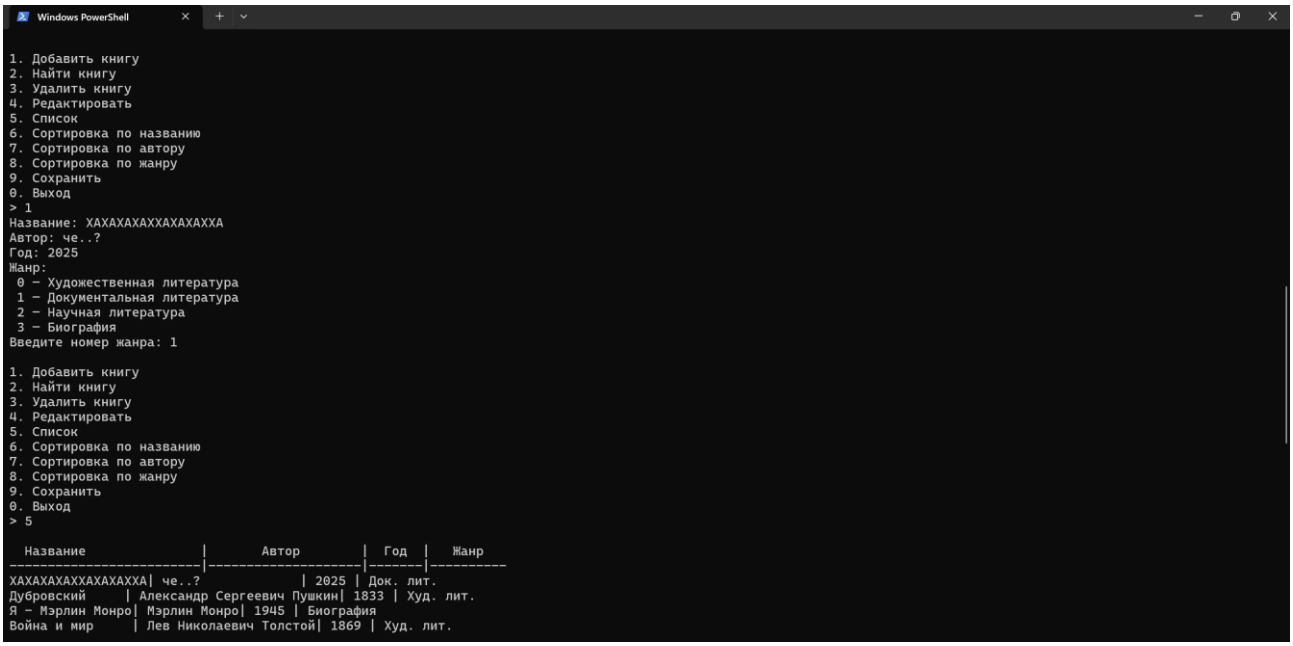
main.cpp

```
#include "Library.h"
#include <thread>
#include <Windows.h>
#include <iostream>

void threadSave(Library& lib, const std::string& filename) {
    lib.saveToFile(filename);
}

void threadLoad(Library& lib, const std::string& filename) {
    lib.loadFromFile(filename);
}

void run(Library& lib, const std::string& filename) {
    int choice;
    while (true) {
        std::cout << "\n1. Добавить книгу\n2. Найти книгу\n3. Удалить книгу\n4.
Редактировать\n"
                  << "5. Список\n6. Сортировка по названию\n7. Сортировка по
автору\n8. Сортировка по жанру\n"
                  << "9. Сохранить\n0. Выход\n> ";
        std::cin >> choice;
        std::cin.ignore();

        switch (choice) {
            case 1: {
                std::string title, author;
                int year, genre;
                std::cout << "Название: ";
                std::getline(std::cin, title);
                std::cout << "Автор: ";
                std::getline(std::cin, author);
                std::cout << "Год: "; std::cin >> year;
                std::cout << "Жанр:\n"
                          << " 0 – Художественная литература\n"
                          << " 1 – Документальная литература\n"
                          << " 2 – Научная литература\n"
                          << " 3 – Биография\n"
                          << "Введите номер жанра: ";
                std::cin >> genre;
                lib.addBook(Book(title, author, year, Book::genreFromInt(genre)));
                break;
            }
            case 2: {
                std::string title;
                std::cout << "Название: ";
                std::getline(std::cin, title);
                Book* b = lib.findBook(title);
                if (b) std::cout << *b << std::endl;
                else std::cout << "Не найдено\n";
                break;
            }
            case 3: {
                std::string title;
                std::cout << "Название: ";
                std::getline(std::cin, title);
                if (lib.removeBook(title)) std::cout << "Удалено\n";
                else std::cout << "Не найдено\n";
                break;
            }
            case 4: {
                std::string title;
                std::cout << "Название редактируемой книги: ";
                std::getline(std::cin, title);
                lib.editBook(title);
                break;
            }
```

```
            case 5: lib.listBooks(); break;
            case 6: lib.sortByTitle(); break;
            case 7: lib.sortByAuthor(); break;
            case 8: lib.sortByGenre(); break;
            case 9: {
                std::thread t(threadSave, std::ref(lib), filename); t.join();
                std::cout << "Сохранено в " << filename << "\n";
                break;
            }
            case 0: return;
            default: std::cout << "Неверный ввод\n";
        }
    }
}

int main(int argc, char* argv[]) {
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);
    Library lib;
    std::string filename;

    try {
        if (argc > 1 && std::string(argv[1]) == "--file") {
            std::cout << "Введите имя файла для загрузки: ";
            std::getline(std::cin, filename);
            std::thread t(threadLoad, std::ref(lib), filename); t.join();
            lib.listBooks();
        } else if (argc > 1 && std::string(argv[1]) == "--hands") {
            std::cout << "Введите имя файла для сохранения: ";
            std::getline(std::cin, filename);
            run(lib, filename);
            std::thread t(threadSave, std::ref(lib), filename); t.join();
        } else {
            std::cout << "Введите имя файла для работы (чтение/сохранение): ";
            std::getline(std::cin, filename);
            std::thread t(threadLoad, std::ref(lib), filename); t.join();
            run(lib, filename);
            std::thread t2(threadSave, std::ref(lib), filename); t2.join();
        }
    } catch (const std::exception& e) {
        std::cerr << "Ошибка: " << e.what() << std::endl;
    }

    return 0;
}
```

Вывод программы





Вывод:

Написал рабочую программу с интерфейсом.