**Федеральное государственное бюджетное профессиональное образовательное учреждение**

**Торжокский политехнический колледж**

**Федерального агентства по государственным резервам**

**ОТЧЁТ**

по итоговой работе за I семестр

**по профессиональному модулю МДК.01.01**

«**РАЗРАБОТКА ПРОГРАМНЫХ МОДУЛЕЙ**»

наименование профессионального модуля

Специальность ***09.02.07 «Информационные системы и программирование»***

Период с «28» ноября 2024г. по «17» апреля 2025г.

Выполнили студенты

(ФИО, подпись)

|  |  |
| --- | --- |
| Преподаватель  Бойков А.М  (ФИО, подпись) | оценка |

2025

**СОДЕРЖАНИЕ**

# **Введение**

Цель работы – освоить принципы командной разработки и сформировать навыки проектирования модульной архитектуры программного обеспечения.

**Задачи:**

1. Определить практическую значимость разрабатываемого программного средства (ПС) для оптимизации рабочих процессов, а также его потенциал в качестве компонента для интеграции в более сложные системы;
2. Обосновать функциональные требования к ПС, включая его роль в решении задач целевой аудитории;
3. Разработать модульную структуру ПС, отражающую взаимодействие классов, их атрибутов и методов, с акцентом на гибкость и масштабируемость;
4. Описать ключевые алгоритмы, реализованные в методах основных классов;
5. Реализовать работоспособный прототип программного средства в среде Visual Studio с использованием языка C# и современных инструментов разработки.

**Сложности:**

* Работа с файлами – нужно корректно загружать и сохранять книги, чтобы не потерять данные.
* Простота использования – интерфейс должен быть интуитивным, даже для преподавателей без технических навыков.

# **Раздел 1. Практическая значимость разрабатываемого ПС**

Данная программа разрабатывается для управления библиотекой книг. Она позволяет автоматизировать процессы добавления, удаления и поиска книг, а также отслеживать, какие книги выданы и кому. Это помогает библиотекарям эффективно управлять библиотечным фондом и предоставлять быстрый и удобный доступ к литературе для пользователей библиотеки.

Если бы не было данной программы все операции выполнялись бы вручную. Это потребовало бы значительных временных затрат и повышало риск ошибок. Без централизованной базы данных поиск нужных книг занимал бы гораздо больше времени, особенно в больших библиотеках с обширным фондом. Также ведение записей вручную сопряжено с риском утраты данных из-за человеческих ошибок, случайного стирания или повреждения бумажных носителей.

В данный момент процесс добавления новой книги происходит следующим образом:

* библиотекарь вручную вписывает данные о книге в журнал или базу данных.

Процесс удаления книги происходит следующим образом:

* библиотекарю необходимо найти книгу в журнале или базе данных и вручную удалить ее.

Процесс поиска книги по автору происходит следующим образом:

* библиотекарь должен просмотреть все записи о книгах вручную, чтобы найти книгу нужного автора.

Процесс записи книги на человека:

* библиотекарь вручную вписывает книгу в карточку читателя.

Процесс возврата книги:

* библиотекарь вручную вычёркивает книгу из карточки читателя.

После введения программы данные процессы станут автоматизированными. Библиотекарь сможет вводить данные через интерфейс программы. В свою очередь программа сама сохраняет и обрабатывает все данные, позволяет моментально выводить данные по запросу пользователя. Благодаря данной программе сократится количество человеко-часов на 50%.

# **Раздел 2. Функциональные требования к ПС**

Основные функции ПС:

1. Добавление новых книг:

Пользователь может ввести информацию о новой книге, включая ее название, автора, издательство, год издания и количество экземпляров. Это позволяет легко пополнять библиотечный фонд новыми книгами.

Чтобы добавить новую книгу, необходимо будет выполнить следующие шаги:

1. Ввести 1 в основном меню для выбора опции «Добавить книгу»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести название книги;
4. Подтвердить нажав клавишу «Enter»;
5. Ввести автора;
6. Подтвердить нажав клавишу «Enter»;
7. Ввести издательство;
8. Подтвердить нажав клавишу «Enter»;
9. Ввести год издания;
10. Подтвердить нажав клавишу «Enter»;
11. Ввести количество экземпляров;
12. Подтвердить создание новых экземпляров книги нажав клавишу «Enter»;
13. На экране появится сообщение «Добавлено … экземпляров книги.».

Название книги, автор и издательство должно вводиться в соответствии с кодировкой UTF8. Год издания и количество экземпляров должны вводится арабскими цифрами.

Если пользователь введёт год издания или количество экземпляров буквами, то на экране появится сообщение «Год издания и количество экземпляров должны вводиться числами.».

Если пользователь введёт отрицательный год издания или количество экземпляров, то на экране появится сообщение «Год издания и количество экземпляров должны быть больше нуля.».

1. Удаление книг:

Если какая-то книга устарела или была заменена новой версией, пользователь может удалить её из базы данных. Это помогает поддерживать актуальность библиотечного фонда.

Чтобы удалить книгу, необходимо будет выполнить следующие шаги:

1. Ввести 2 в основном меню для выбора опции «Удалить книгу»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести ID книги;
4. Подтвердить удаление книги нажав клавишу «Enter»;
5. На экране появится сообщение «Книга '…' (ID: …) удалена.».

Если книги с таким ID нет в библиотеке, то на экране появится сообщение «Книга не найдена.».

Если пользователь введёт ID буквами, то на экране появится сообщение «ID должно вводиться числом.».

Если пользователь введёт отрицательный ID, то на экране появится сообщение «ID должно быть больше нуля.».

ID должно вводится арабскими цифрами.

1. Поиск книг:

Программа предоставляет возможность искать книги по автору и по названию книги (частичному совпадению). Это особенно полезно для пользователей библиотеки, которые могут быстро найти нужную литературу.

Чтобы осуществить поиск книг по автору, необходимо будет выполнить следующие шаги:

1. Ввести 3 в главном меню для выбора опции «Поиск по автору»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести автора для поиска;
4. Подтвердить поиск книги нажав клавишу «Enter»;
5. Программа покажет все подходящие книги.

Если книг данного автора нет в библиотеке, то на экране появится сообщение «Книги не найдены.».

Если пользователь введёт пустого автора, то программа покажет все книги.

Автор книги должен вводиться в соответствии с кодировкой UTF8.

Чтобы осуществить поиск книг по названию книги (частичному совпадению), необходимо будет выполнить следующие шаги:

1. Ввести 4 в основном меню для выбора опции «Поиск по названию или частичному совпадению»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести название или несколько первых букв названия книги;
4. Подтвердить поиск книги нажав клавишу «Enter»;
5. Программа покажет все подходящие книги.

Если книги с данным названием нет в библиотеке, то на экране появится сообщение «Книги не найдены.».

Если пользователь введёт пустое название, то программа покажет все книги.

Название книги должно вводиться в соответствии с кодировкой UTF8.

1. Запись книги на человека:

В программе можно указать, кто и когда взял определенную книгу. Это помогает библиотекарю контролировать выдачу и возврат книг, предотвращая их потерю.

Чтобы записать книгу за определенным человеком, необходимо будет выполнить следующие шаги:

1. Ввести 5 в основном меню для выбора опции «Записать книгу»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести ID книги;
4. Подтвердить ID книги нажав клавишу «Enter»;
5. Ввести ФИО человека, который взял книгу;
6. Подтвердить ФИО нажав клавишу «Enter»;
7. На экране появится сообщение «Книга '…' записана за '…'.».

Если книги с таким ID нет в библиотеке, то на экране появится сообщение «Книга не найдена.».

Если пользователь введёт ID буквами, то на экране появится сообщение «ID должно вводиться числом.».

Если пользователь захочет взять уже записанную книгу, то на экране появится сообщение «Книга уже взята.».

ID должно вводится арабскими цифрами.

ФИО должно вводиться в соответствии с кодировкой UTF8.

1. Возврат книги:

Чтобы вернуть книгу, необходимо будет выполнить следующие шаги:

1. Ввести 6 в основном меню для выбора опции «Вернуть книгу»;
2. Подтвердить нажав клавишу «Enter»;
3. Ввести ID книги;
4. Подтвердить ID книги нажав клавишу «Enter»;
5. На экране появится сообщение «Книга возвращена.».

Если книги с таким ID нет в библиотеке или она не взята, то на экране появится сообщение «Книга не найдена или не была взята.».

ID должно вводится арабскими цифрами.

Если пользователь введёт ID буквами, то на экране появится сообщение «ID должно вводиться числом.».

ID должно вводится арабскими цифрами.

1. Выход из программы:

Чтобы выйти из программы, необходимо будет выполнить следующие шаги:

1. Ввести 7 в основном меню для выбора опции «Выход»;
2. Подтвердить нажав клавишу «Enter».

Программа должна проверять, что вводимые данные соответствуют необходимым типам данным (число – int, строка - string).

# **Раздел 3. Модульная структура проектируемого ПС**

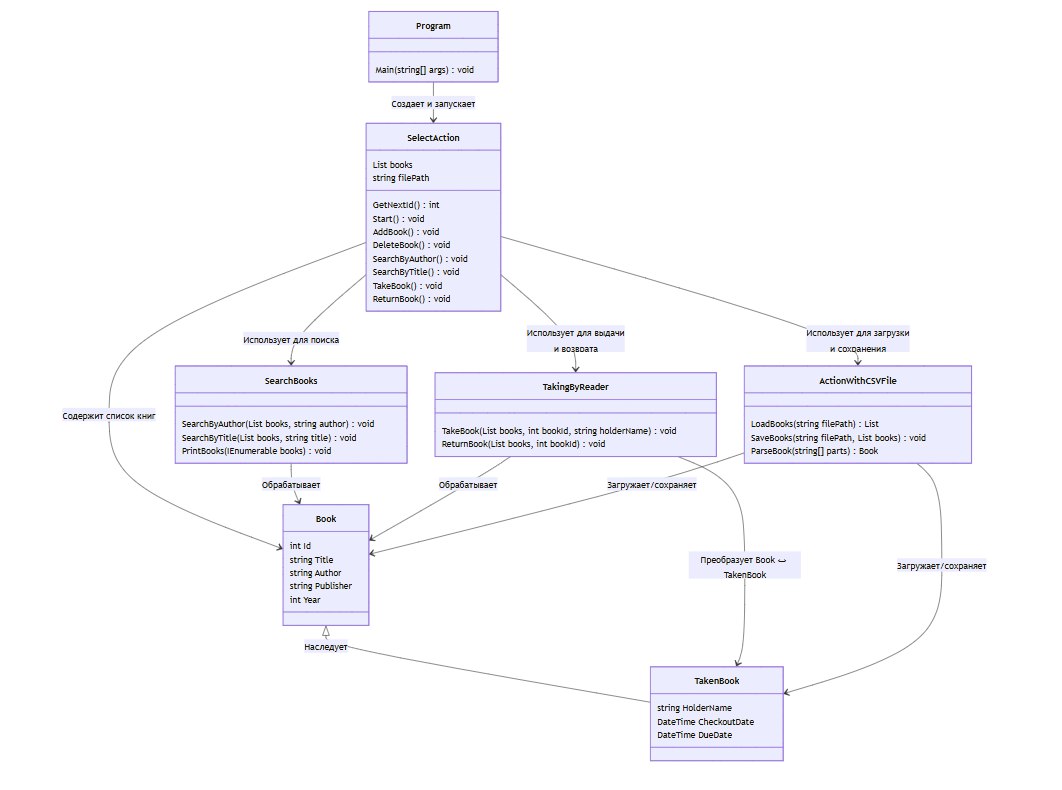


Рисунок 1 - диаграмма классов.

Какие поля или методы класса «Книга» наследует класс «Записанная\_книга»? Связь должна быть содержит или включает, наследования тут никакого нет.

Классы, поля и методы на диаграмме по стандарту именуется на английском языке, названия разделов также на английском

У полей не указаны типы данных

У методов не указаны типы возвращаемых значений

Связи подписаны неверно, должен быть указан глагол или отглагольное существительное, которое показывает, как связаны между собой классы (например, «содержит», «использует метод», «запрашивает данные», «наследует» и т.д.)

**Критерии выделения классов:**

* Избежание дублирования кода - разделение ответственности между классами позволяет избежать дублирования кода и улучшить структурированность программы.
* Легкость поддержки и расширения - чёткое разделение функционала на отдельные классы делает код более понятным и простым в поддержке и расширении.

**Структура классов:**

1. **Класс:** SelectAction

**Данные класса:**

* List<Book> books — список всех книг в библиотеке (включая TakenBook).
* string filePath — путь к файлу, в котором хранятся данные о книгах.
* GetNextId() — генерирует уникальный ID для новой книги.

**Методы класса:**

* AddBook() — добавляет новую книгу (или несколько экземпляров) в список.
* DeleteBook() — удаляет книгу по ID.
* SearchByAuthor() — передаёт ввод пользователя в метод поиска по автору.
* SearchByTitle() — передаёт ввод пользователя в метод поиска по названию.
* TakeBook()— вызывает метод взятия книги читателем.
* ReturnBook() — вызывает метод возврата книги.
* Start() — запускает цикл меню для взаимодействия с пользователем.

**Описание:**

Основной класс, управляющий взаимодействием пользователя с программой через консольный интерфейс. Отвечает за отображение меню, обработку ввода и координацию работы других классов (добавление, удаление, поиск книг, а также их выдачу и возврат). Обеспечивает загрузку данных при запуске и сохранение изменений при завершении работы.

1. **Класс:** Book

**Данные класса:**

* int Id — уникальный идентификатор.
* string Title — название.
* string Author — автор.
* string Publisher — издательство.
* int Year — год издания.

**Описание:**

Базовый класс, представляющий книгу в библиотеке. Содержит общие данные о книге (название, автор, издательство, год издания) и служит основой для всех операций. Используется для книг, доступных для выдачи.

1. **Класс:** TakenBook

**Данные класса:**

* string HolderName — ФИО читателя.
* DateTime CheckoutDate — дата взятия.
* DateTime DueDate — срок возврата.

**Описание:**

Класс-наследник Book, описывающий книгу, находящуюся у читателя. Хранит информацию о том, кто взял книгу, дату выдачи и срок возврата. При сохранении в CSV добавляет дополнительные поля, а при возврате преобразуется обратно в обычную книгу (Book), удаляя данные о читателе.

1. **Класс:** ActionWithCSVFile

**Методы класса:**

* LoadBooks() — загружает данные из CSV-файла, преобразуя строки в объекты Book или TakenBook.
* SaveBooks() — сохраняет список книг в CSV-файл, включая статус выдачи.
* ParseBook() — парсит строку CSV в объект книги.

**Описание:**

Класс, отвечающий за чтение и запись данных о книгах в CSV-формате. Преобразует объекты Book и TakenBook в строки CSV и обратно, обеспечивая долговременное хранение данных. Автоматически определяет статус книги (выдана или нет) при загрузке.

Формат CSV:

* Для обычной книги: Id,Title,Author,Publisher,Year,,,.
* Для взятой книги: Id,Title,Author,Publisher,Year,HolderName,CheckoutDate,DueDate.

**Особенности CSV:**

* При чтении файла в Excel могут возникать ошибки кодировки (например, кириллица отображается некорректно). Это не влияет на работу программы.
* Пустые поля для обычных книг заполняются запятыми без данных.

1. **Класс:** SearchBooks

**Методы класса:**

* SearchByAuthor() — ищет книги по частичному совпадению автора.
* SearchByTitle() — ищет книги по частичному совпадению названия.
* PrintBooks() — выводит информацию о книгах, включая статус выдачи.

**Описание:**

Класс, реализующий логику поиска книг по автору или названию. Фильтрует список книг на основе введенных пользователем критериев и выводит результаты, включая информацию о статусе выдачи (если книга находится у читателя).

1. **Класс:** TakingByReader

**Методы класса:**

* TakeBook() — преобразует Book в TakenBook, устанавливая даты и ФИО читателя.
* ReturnBook() — преобразует TakenBook обратно в Book.

**Описание:**

Класс, управляющий операциями выдачи и возврата книг. Преобразует обычную книгу (Book) в выданную (TakenBook) при взятии, устанавливая даты и ФИО читателя, и выполняет обратное преобразование при возврате.

1. **Класс:** Program

**Методы класса:**

* Main() — создаёт экземпляр SelectAction и запускает программу.

**Описание:**

Класс является точкой входа в программу. Он инициирует работу приложения. Не содержит бизнес-логики, только инициализирует работу программы.

**Взаимосвязь классов программы:**

Класс Program является точкой входа в приложение. Создаёт экземпляр SelectAction и запускает его метод Start(), передавая управление основному меню.

Класс SelectAction выступает центральным координатором:

* Использует ActionWithCSVFile для загрузки данных при запуске (LoadBooks()) и сохранения при завершении (SaveBooks()).
* Хранит список книг (List<Book>), включая как обычные книги (Book), так и выданные (TakenBook).
* Вызывает методы TakingByReader для выдачи (TakeBook()) и возврата (ReturnBook()) книг.
* Перенаправляет запросы поиска в SearchBooks (SearchByAuthor(), SearchByTitle()).
* Управляет пользовательским вводом: добавляет/удаляет книги, обрабатывает ошибки ввода.

Класс Book и TakenBook:

* Book хранит базовую информацию о книге (ID, название, автор и т.д.).
* TakenBook наследует от Book и добавляет поля:

HolderName — ФИО читателя.

CheckoutDate и DueDate — даты выдачи и возврата.

* Связь с другими классами:

Объекты Book и TakenBook хранятся в списке SelectAction.

ActionWithCSVFile преобразует их в строки CSV и обратно.

SearchBooks проверяет тип книги при выводе результатов.

TakingByReader заменяет Book на TakenBook при выдаче и обратно при возврате.

Класс ActionWithCSVFile отвечает за работу с данными:

* Загружает данные из CSV:

Парсит строки в объекты Book или TakenBook в зависимости от наличия HolderName.

* Сохраняет данные в CSV:

Для TakenBook записывает дополнительные поля (HolderName, даты).

Для Book оставляет пустые колонки.

Класс SearchBooks реализует логику поиска:

* Использует список книг из SelectAction.
* Фильтрует книги по автору (SearchByAuthor()) или названию (SearchByTitle()).
* Выводит результаты, проверяя тип книги:

Для TakenBook добавляет информацию о читателе и датах.

Класс TakingByReader управляет статусом книг:

* Выдача книги:

Проверяет, не выдана ли книга (если Book → заменяет на TakenBook).

Устанавливает HolderName, CheckoutDate, DueDate.

* Возврат книги:

Проверяет, выдана ли книга (если TakenBook → заменяет на Book).

Удаляет данные о читателе.

##################################################################

Написанное выше соответствует запросам, но может быть неправильно оформлено (интервал)

**Раздел 4. Описание алгоритмов программы**

1. **Метод добавления новой книги.**

**Шаги алгоритма:**

1. **Начало**
   * + - Алгоритм запускается при инициации процесса добавления книги.
2. **Ввод данных пользователем**
   * + - Программа запрашивает у пользователя:

Название книги,

Автора,

Издательство,

Год издания,

Количество экземпляров.

1. **Первая проверка: числовой формат**
   * + - Проверяется, являются ли введенные значения для года издания и количества экземпляров числами.

Если нет:

Выводится ошибка: *«Год издания и количество экземпляров должны вводиться числами»*.

Алгоритм завершается.

Если да:

Переход к следующей проверке.

1. **Вторая проверка: значения больше нуля**
   * + - Проверяется, что год издания и количество экземпляров больше нуля.

Если нет:

Выводится ошибка: *«Год издания и количество экземпляров должны быть больше нуля»*.

Алгоритм завершается.

Если да:

Переход к генерации ID.

1. **Генерация ID**
   * + - Система создает уникальный идентификатор для книги.
2. **Проверка ID на числовой формат**
   * + - Проверяется, является ли сгенерированный ID числом.

Если нет:

Выводится ошибка: *«ID должно вводиться числом»*.

Алгоритм завершается.

Если да:

Переход к добавлению книги.

1. **Добавление книги в систему**
   * + - Книга сохраняется в базе данных с присвоенным ID.
       - Выводится сообщение: *«Добавлено [n] количество экземпляров»*, где [n]— введенное количество.
2. **Конец**
   * + - Алгоритм завершает работу.

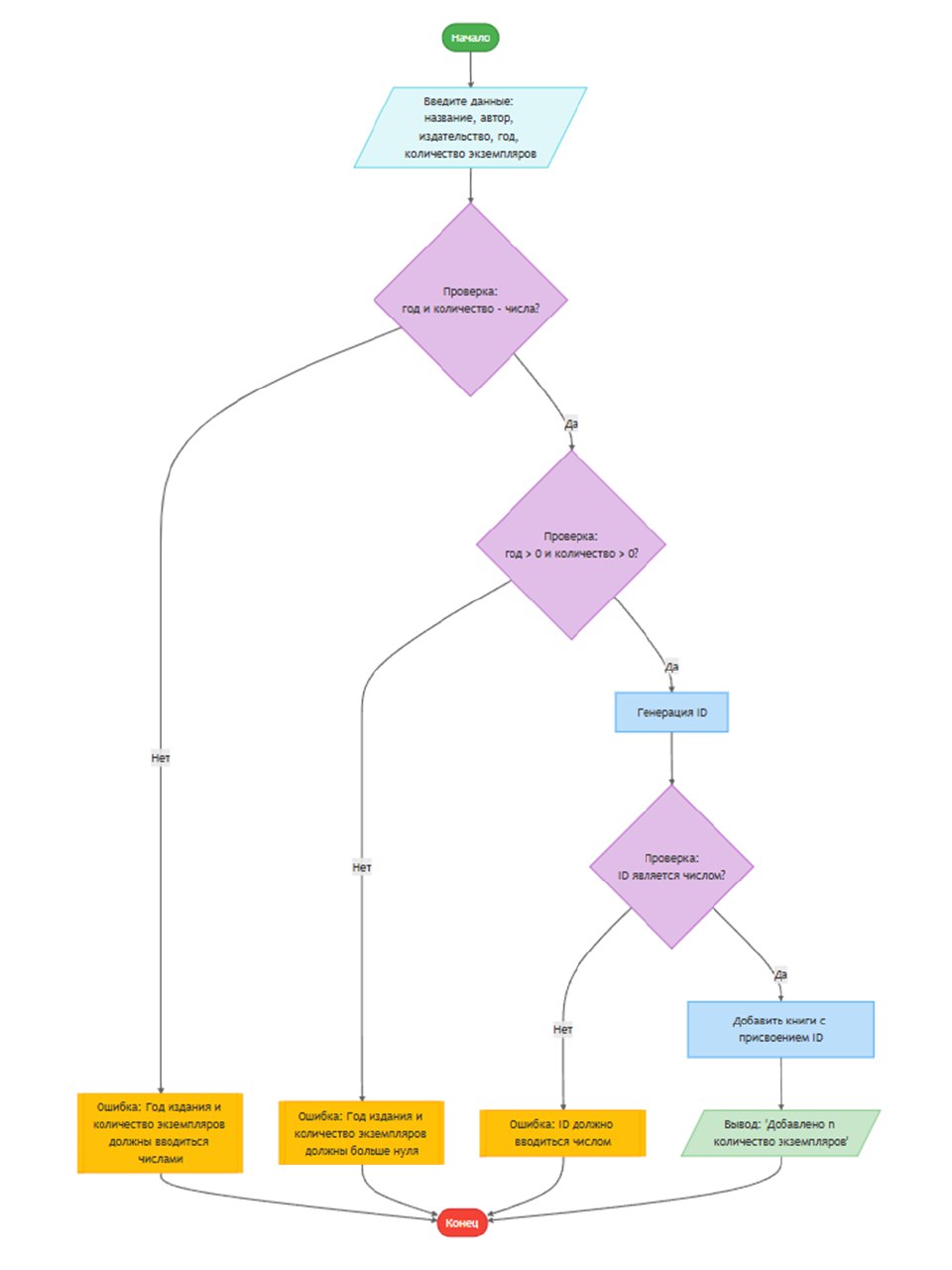


Рисунок 2 - блок-схема для метода добавления новой книги.

1. **Метод поиска книги по автору.**

**Шаги алгоритма:**

1. **Начало**
   * + - Алгоритм запускается при инициации процесса поиска книг.
2. **Ввод автора**
   * + - Пользователю предлагается ввести имя автора через интерфейс программы.
3. **Проверка наличия автора**
   * + - Проверяется, был ли введен автор:

Если да:

Запускается процесс «Поиск книг по автору».

Если нет:

Запускается процесс «Получить все книги».

1. **Сценарий 1: Поиск по автору**
   * + - Система ищет книги в базе данных, используя введенное имя автора.
       - Проверка результата поиска:

Если книги найдены:

Выводится список найденных книг.

Если книги не найдены:

Выводится сообщение: *«Книги не найдены»*.

1. **Сценарий 2: Получение всех книг**
   * + - Система извлекает полный список книг из базы данных.
       - Выводится список всех доступных книг.
2. **Конец**
   * + - Алгоритм завершает работу после вывода результатов.

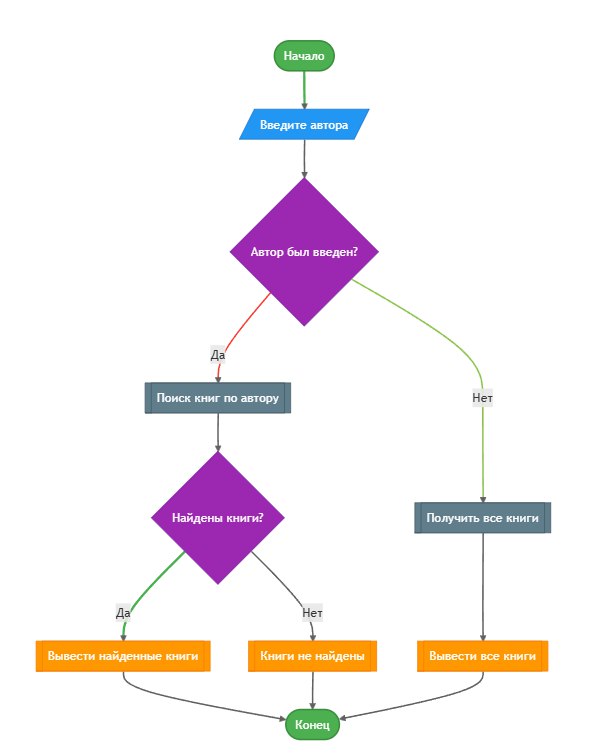


Рисунок 3 – блок-схема для метода поиска книг по автору.

# **Раздел 5.** Разработка проекта Visual Studio C#

Рисунок 6 - Точка входа в программу



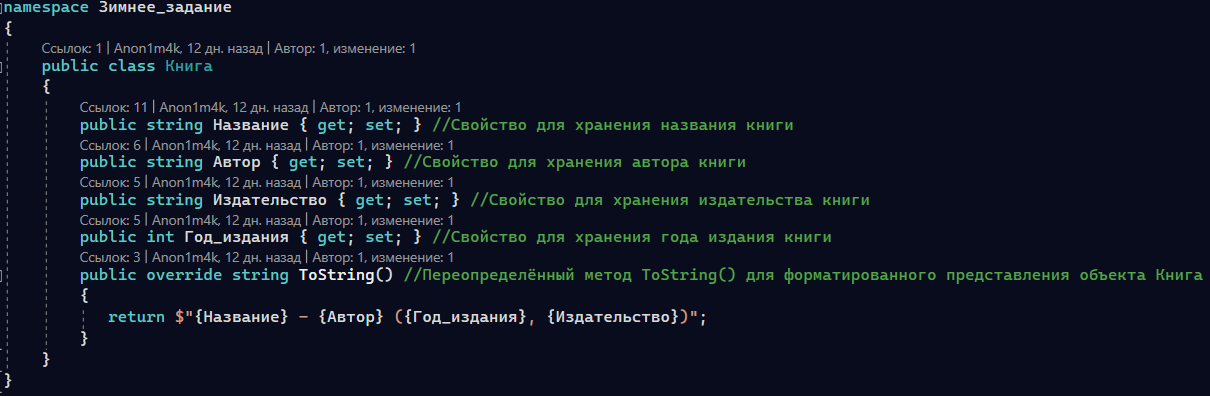
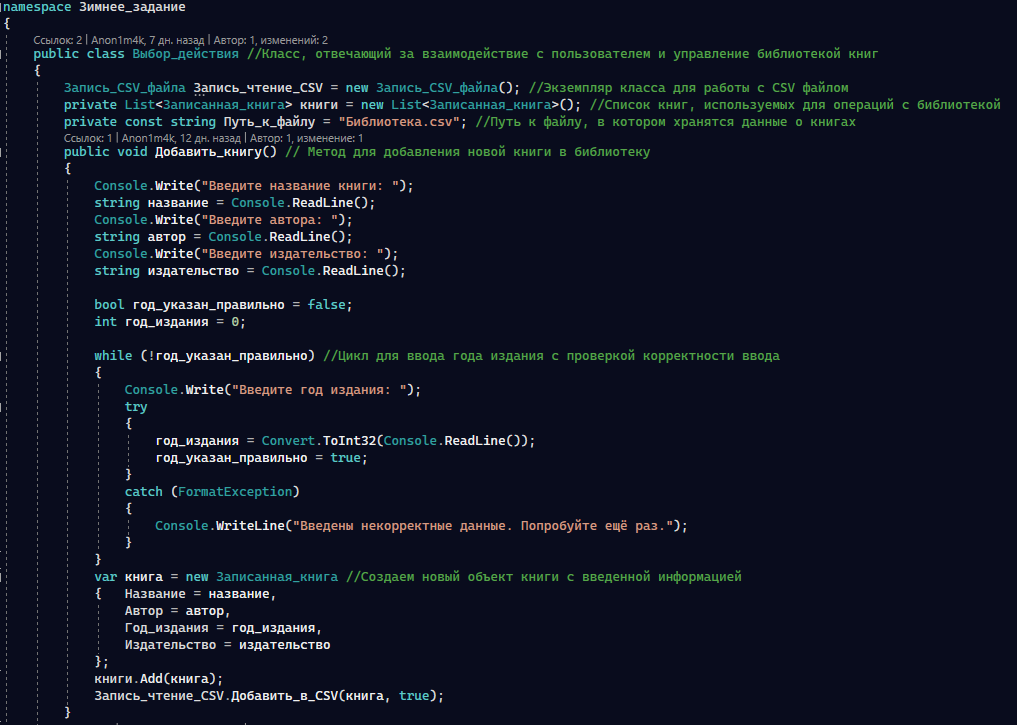
Рисунок 8 – Методы для записи и возврата книги

Рисунок 9 - Определение класса "Книга"

Рисунок 10 – Метод добавления новой книги

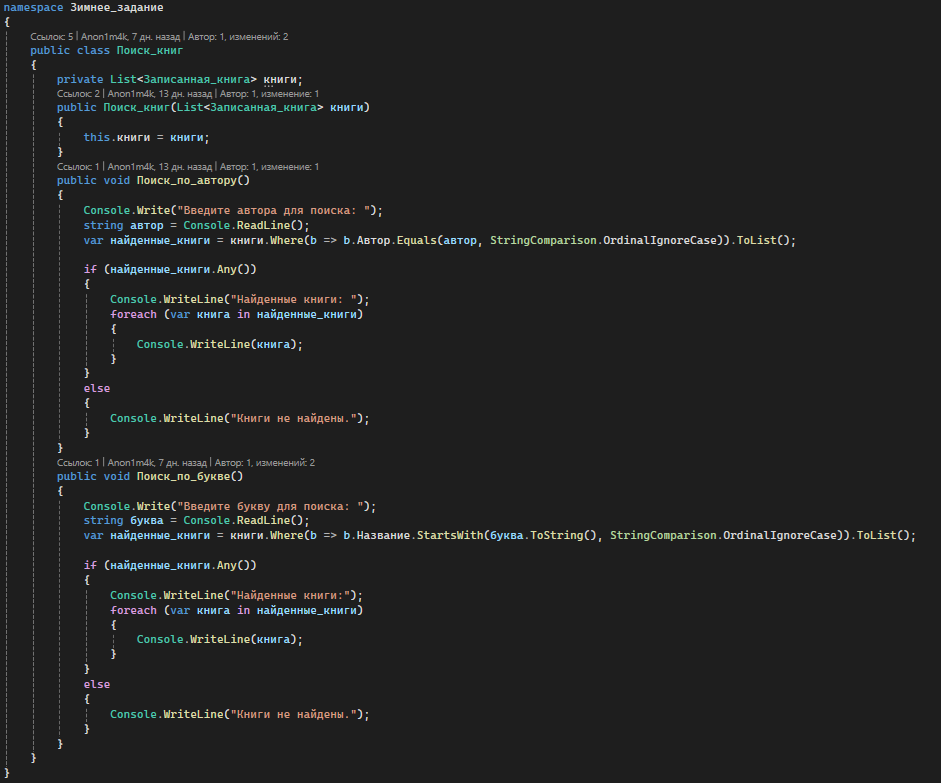


Рисунок 11 – Методы для поиска книг по букве и по автору

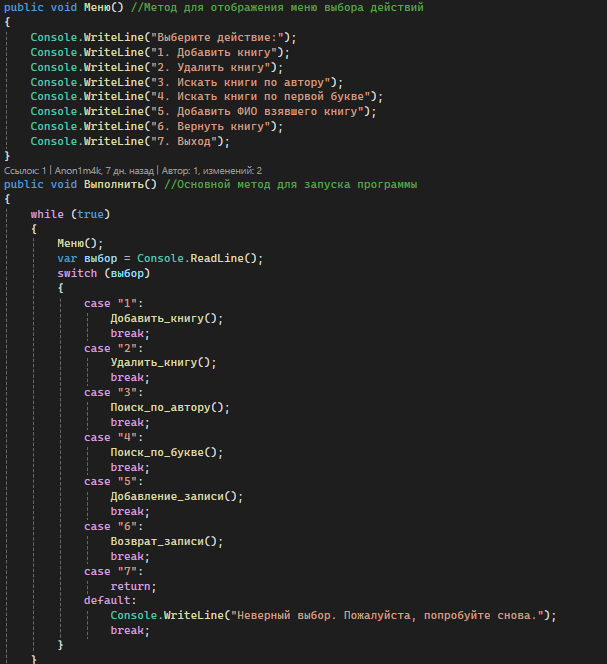
****

Рисунок 12 - Метод меню для выбора действия

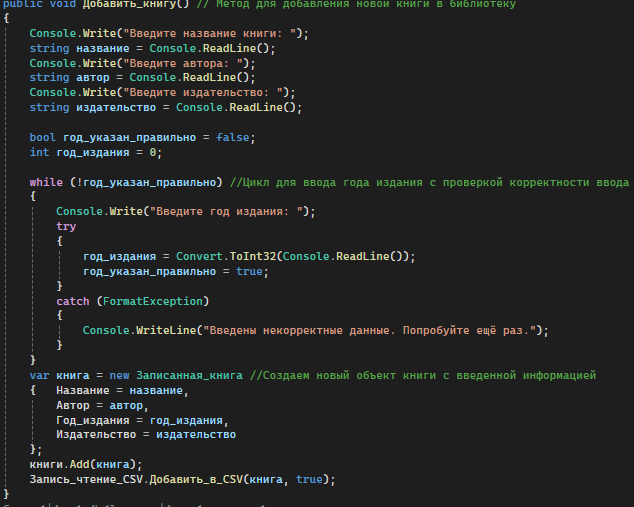


Рисунок 13 - Метод добавления новой книги

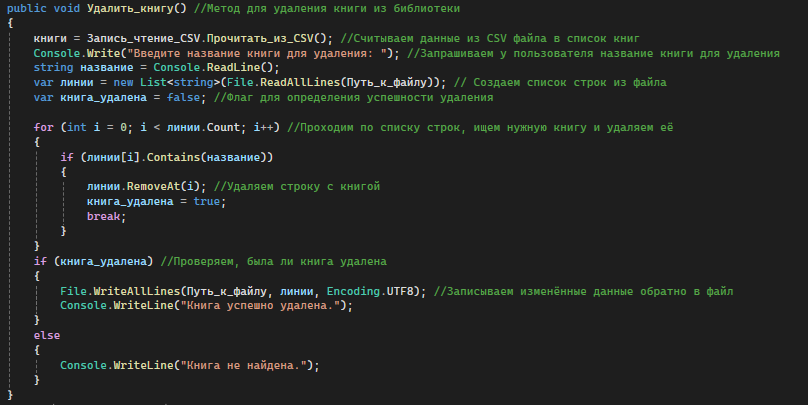


Рисунок 14 – Метод для удаления книги

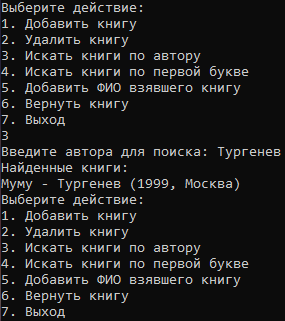


Рисунок 1 – Пример поиска книги по автору

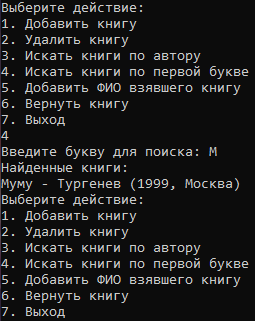


Рисунок 2 – Пример поиска книги по первой букве