**СОДЕРЖАНИЕ**

# **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.

Здесь необходимо привести также типичные сценарии (примеры), как происходят процессы сейчас, и каким образом данные процессы могут быть облегчены с использованием программы.

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

Основные функции программы:

* Добавление новых книг:

Пользователь может ввести информацию о новой книге, включая ее название, автора, издательство и год выпуска. Это позволяет легко пополнять библиотечный фонд новыми поступлениями.

Необходимо дополнить требование описанием, каким образом необходимо организовать ввод данных, а также предусмотреть проверку вводимой информации

* 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.

Необходимо дополнить требование описанием, каким образом пользователь будет производить удаление, на основе каких реквизитов, а также предусмотреть информационные сообщения:

- подтверждение удаления книги;

- успешность удаления или неудача

* Поиск книг:

Программа предоставляет возможность искать книги по автору или по первой букве названия. Это особенно полезно для пользователей библиотеки, которые могут быстро найти нужную литературу.

Дополнить требование поиском по точному названию книги

Описать требования к формату выходной информации в ответ на запрос пользователя

Описать форматы возможных запросов поиска пользователем

* Отслеживание выданных книг:

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

Описать требования к процессу выдачи и возврата книги.

Предусмотреть реакцию программы на следующие ситуации:

- запрашиваемая книга выдана;

- запрашиваемая книга отсутствует;

- книга успешно возвращена;

- формат отчета о выданных книгах

**Примеры использования. (соотнести сразу в соответствующем пункте описания требований)**

**Пример 1: Добавление новой книги**

Чтобы добавить новую книгу, выполните следующие шаги:

1. Введите 1 в основном меню для выбора опции "Добавить книгу".
2. Введите название книги, автора, год издания и издательство.
3. Программа сохранит данные и вернет вас в главное меню.

**Пример 2: Поиск книги по автору**

Чтобы найти книги определенного автора, выполните следующие шаги:

1. Введите 3 в главном меню для выбора опции "Искать книги по автору".
2. Введите имя автора, которого хотите найти.
3. Программа покажет все подходящие книги.

**Пример 3: Удаление книги**

1. Введите 2 в основном меню для выбора опции «Удалить книгу».
2. Введите название книги.
3. Книга будет удалена.

**Пример 4: Поиск книги по первой букве**

1. Введите 4 в основном меню для выбора опции «Искать книги по первой букве»
2. Введите первую букву названия книги
3. Программа покажет все подходящие книги

**Пример 5: Добавление фамилии взявшего книгу**

1. Введите 5 в основном меню для выбора опции «Добавление фамилии взявшего книгу»
2. Введите название книги
3. Введите фамилию взявшего книгу

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

**Раздел должен начинаться диаграммой классов!!!**

**Описание методов и полей приводить в сигнатуре C#!!!** Сигнатура метода представляет собой тип возвращаемого значения, имя и список аргументов, сигнатура поля – тип и имя. Сигнатуру классов (их полей и методов) отобразить на диаграмме классов, а также их связи.

**Класс:** Выбор\_действия

Данные класса:

Запись\_чтение\_CSV (Запись\_CSV\_файла) (тут лучше оставить просто Запись\_CSV\_файла) — ссылка на экземпляр класса Запись\_CSV\_файла.

книги (List<Книга>) — список всех книг в библиотеке.

Методы класса:

Добавить\_книгу() — добавляет новую книгу в список книг.

Удалить\_книгу() — удаляет книгу из списка книг.

Поиск\_по\_автору() — выполняет поиск книг по автору.

Поиск\_по\_букве() — выполняет поиск книг по первой букве названия.

Добавление\_записи() — обновляет информацию о том, кто взял конкретную книгу.

Меню() — отображает меню выбора действий.

Выполнить() — запускает цикл взаимодействия с пользователем.

Описание:

Этот класс представляет собой интерфейс пользователя. Он взаимодействует с пользователем, предлагая различные действия (например, добавить или удалить книгу), и вызывает соответствующие методы других классов для выполнения этих действий. Данный класс важен, так как он обеспечивает взаимодействие пользователя с системой и управляет основными процессами работы с библиотекой.

**Класс:** Книг~~и~~а

Данные класса:

Название (string) — название книги.

Автор (string) — имя автора книги.

Издательство (string) — издательство, выпустившее книгу.

Год\_издания (int) — год издания книги.

Записано\_за (string) — фамилия человека, который взял книгу.

Данное поле не является характеристикой любой книги. Все выписанные книги правильно хранить в рамках отдельного класса.

Методы класса:

ToString() — переопределённый метод для представления объекта книги в виде строки.

**Описание:**

Класс Книга является основным классом в системе. Каждый объект этого класса содержит информацию о книге, такую как её название, автор, издательство, год издания и текущий статус (кто взял книгу). Этот класс необходим, так как он является основой для всех операций с книгами в библиотеке.

**Класс:** Запись\_CSV\_файла

**В требованиях не описано, для чего нужен csv-файл, какого он должен быть формата, какие могут быть связанные с ним ошибки и реакция программы на эти ошибки, а также требуется привести примеры такого csv-файла.**

Данные класса:

Путь\_к\_файлу (string, статическое поле ??? (необходимо обоснование, почему это поле должно быть статическое)) — путь к CSV-файлу, где хранятся записи о книгах.

Методы класса:

Сохранить\_в\_CSV(List<Книга>) — сохраняет список книг в CSV-файл.

Добавить\_записть\_в\_CSV(List<Книга>) — добавляет новую запись о книге в CSV-файл.

Прочитать\_из\_CSV() — читает содержимое CSV-файла и преобразует его в список книг.

**Описание:**

Класс Запись\_CSV\_файла отвечает за взаимодействие с CSV-файлом, в котором хранятся данные о книгах. Он предоставляет методы для сохранения списка книг в файл, добавления новых записей и чтения данных из файла. Этот класс важен, так как он обеспечивает долгосрочное хранение данных о книгах и их восстановление при необходимости.

**Класс:** Program

Данные класса:

Instance (Выбор\_действия) — экземпляр класса Выбор\_действия.

Методы класса:

Main(string[]) — главный метод программы, который запускает приложение и вызывает метод Выбор\_действия.Выполнить().

**Описание:**

Класс Program является точкой входа в программу. Он создаёт экземпляр класса Выбор\_действия и запускает его метод Выполнить (), который начинает взаимодействие с пользователем. Этот класс необходим, так как он инициализирует работу приложения и передаёт управление классу Выбор\_действия.

Необходимо выделить класс, который будет фиксировать факты выдачи и возврата книг

Необходимо выделить класс, который будет реализовывать операции поиска книг по запросам пользователям

**Критерии выделения классов:**

1. **Каждый класс имеет чётко определённую ответственность**:

* Класс Книга отвечает за представление одной книги и хранение всей необходимой информации о ней.
* Класс Запись\_CSV\_файла отвечает за взаимодействие с CSV-файлом и хранение данных о книгах.
* Класс Выбор\_действия отвечает за взаимодействие с пользователем и выполнение основных операций с книгами.

1. **Избежание дублирования кода**:

* Разделение ответственности между классами позволяет избежать дублирования кода и улучшить структурированность программы.

1. **Легкость поддержки и расширения**:

* Чёткое разделение функционала на отдельные классы делает код более понятным и простым в поддержке и расширении.

**Связи между классами:**

Экземпляр Выбор\_действия использует экземпляр Запись\_CSV\_файла:

* Он создаёт и хранит ссылку на экземпляр Запись\_CSV\_файла, чтобы использовать его методы для работы с данными.

Экземпляр Запись\_CSV\_файла работает с экземпляром Книга:

* Он принимает список объектов Книга через свои методы и манипулирует ими для сохранения и чтения данных.

Экземпляр Выбор\_действия работает с экземпляром Книга:

* Он создает объекты Книга и добавляет их в список книги, который затем используется для взаимодействия с Запись\_CSV\_файла.

**Аргументация выбора типов полей:**

* Тип поля string для Название, Автор, Издательство и Записано\_за:

Эти поля содержат текстовую информацию, которая может быть разной длины и формата. Тип string подходит для хранения таких данных.

* Тип поля int для Год\_издания:

Год издания книги всегда представлен числовым значением, поэтому использование типа int логично и удобно для обработки и хранения.

* Статическое поле Путь\_к\_файлу:

Это поле должно быть статическим, так как оно определяет местоположение CSV-файла, которое не меняется во время исполнения программы (статическое поле это не гарантирует. Гарантию того, что данные не будут меняться при исполнении программы, определяет const). ~~Статические поля позволяют всем экземплярам класса обращаться к одному и тому же значению~~. Это не нужно для имени файла

**Порядок использования методов:**

1. Создаётся экземпляр Выбор\_действия.
2. Пользователь выбирает действие из меню (например, добавить книгу).
3. Внутри метода Добавить\_книгу() создаётся объект Книга с использованием данных, введённых пользователем.
4. Объект Книга добавляется в список книги.
5. Затем вызывается метод Сохранить\_в\_CSV(List<Книга>), который сохраняет все книги в CSV-файл.

Таким образом, последовательность действий, следующая: создание объекта → ввод данных → сохранение данных.

# **Раздел 4.** Описание алгоритмов

Опишем метод добавления новой книги.

Шаги алгоритма:

1. **Начало**: Начало алгоритма.
2. **Ввод данных о книге**: Пользователь вводит данные о книге (название, автор, издательство, год издания).
3. **Создание объекта Книга**: на основе введенных данных создается новый объект Книга.
4. **Добавление созданного объекта в список книг**: новый объект добавляется в список книги в классе Выбор\_действия.
5. **Вызов метода Сохранить\_в\_CSV (List <Книга>)**: метод Сохранить\_в\_CSV класса Запись\_CSV\_файла используется для сохранения списка книг в CSV-файл.
6. **Вывод сообщения об успешном сохранении книги**: после завершения сохранения выводится сообщение об успешном завершении операции.
7. **Конец**: Конец алгоритма.

**Предусмотреть ветвление на случай неуспешного сохранения данных о книгах в файл**

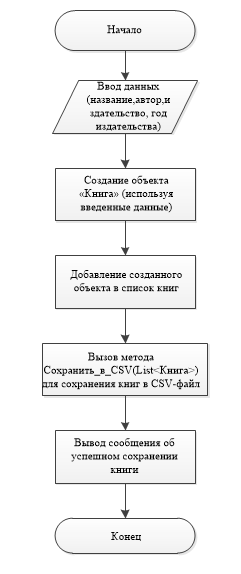


Схема 1 – Блок-схема для метода добавления новой книги

Давайте рассмотрим второй немаловажный метод: поиск книги по автору.

Шаги алгоритма:

1. **Начало**: Начало алгоритма.
2. **Получение ввода от пользователя (ввод автора)**: Пользователь вводит автора, по которому будет осуществлен поиск.
3. **Чтение данных из CSV-файла (метод Прочитать\_из\_CSV())**: Метод Прочитать\_из\_CSV() используется для чтения всех книг из CSV-файла и преобразования их в список объектов Книга.

Данный алгоритм следует выполнить один раз при запуске программы, все данные должны будут просто храниться в памяти в соответствующей структуре и обновляться по ходу взаимодействия пользователя с программой. Так мы снизим накладные расходы на обращение каждый раз к диску за одной и той же почти информацией.

1. **Фильтрация книг по автору (метод Where)**: Where применяется для фильтрации списка книг, оставляя только те, авторы которых совпадают с введенным именем. Предусмотреть ветвление для случая, когда книг указанного автора не найдено
2. **Вывод найденных книг**: Найденные книги выводятся на экран.
3. **Конец**: Конец алгоритма.

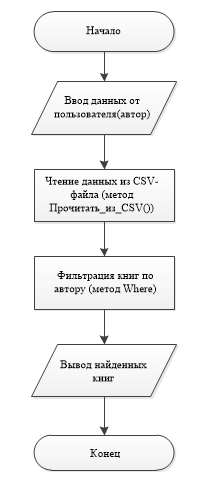


Схема 2 – Метод для поиска книг по автору

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

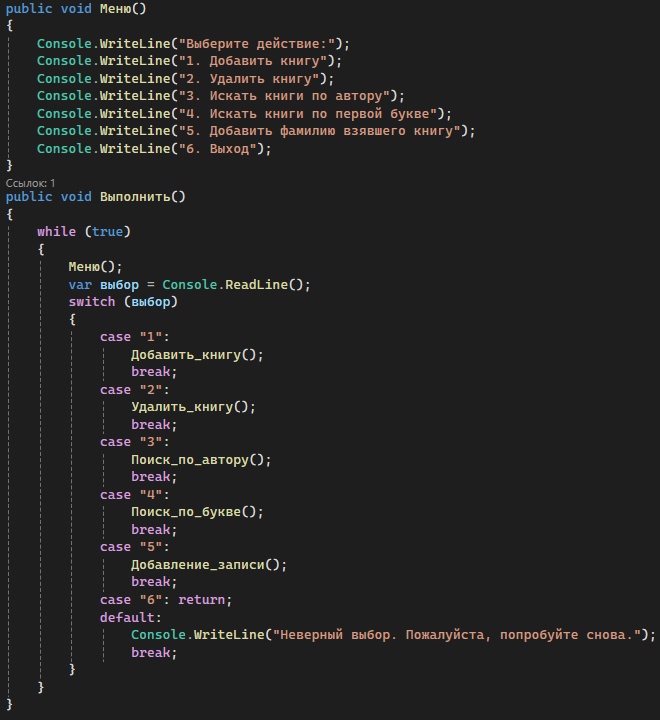


Рисунок 1 – Метод меню для выбора действия

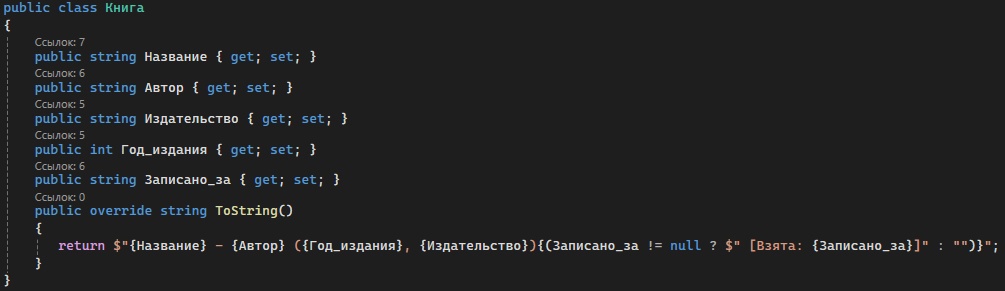


Рисунок 2 – Определение класса "Книга" с определением свойств и переопределением метода ToString

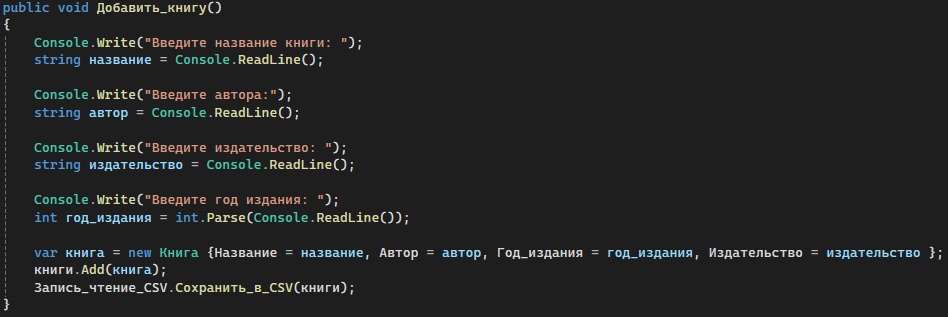


Рисунок 3 - Метод добавления новой книги

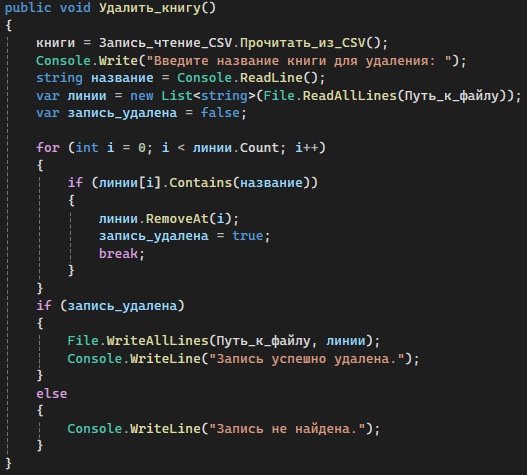


Рисунок 4 – Метод удаления книги

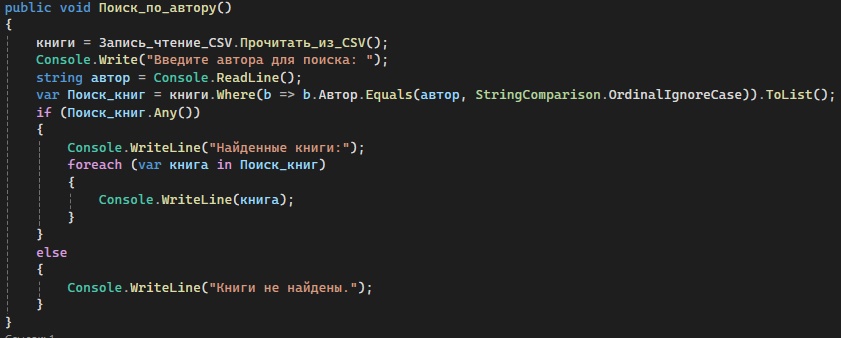


Рисунок 5 – Метод поиска книг по автору



Рисунок 6 – Метод поиска книги по букве

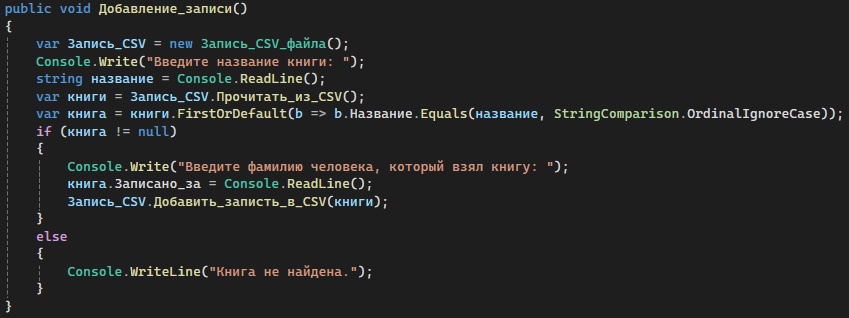


Рисунок 7 - Метод добавления отметки взятия книги человеком

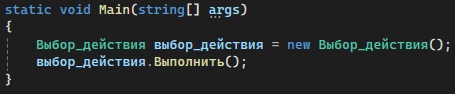


Рисунок 8 – Точка входа в программу

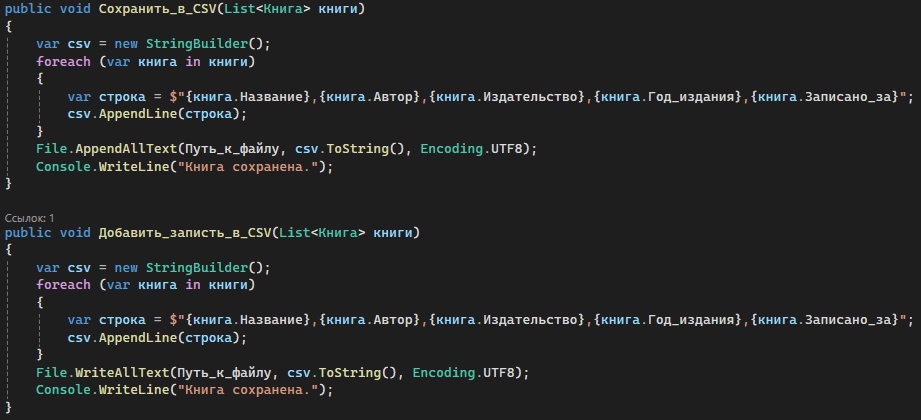


Рисунок 9 - Методы записи информации в CSV файл

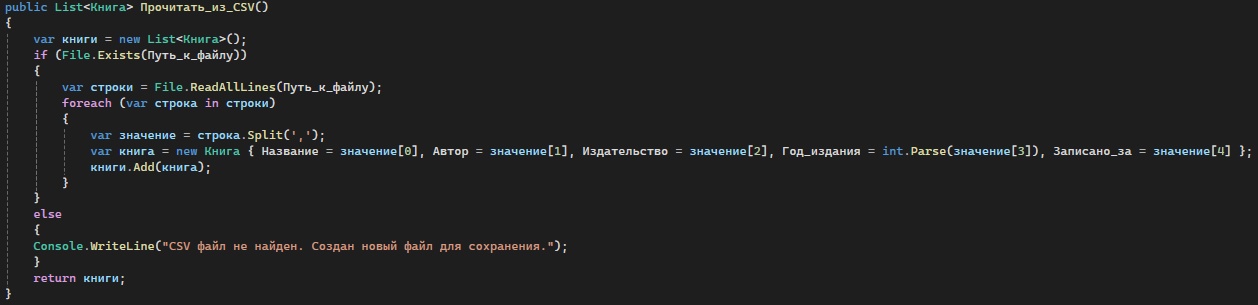


Рисунок 10 - Метод чтения информации из CSV файла