Федеральное государственное бюджетное профессиональное образовательное учреждение

Торжокский политехнический колледж

Федерального агентства по государственным резервам

Индивидуальное задание по МДК.01.01

«Разработка программных модулей»

Выполнили студенты группы П-30

Смирнова Е.В.

Курчаков А.Д.

2025

**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.

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 каталоге и вручную удалить ее.

Процесс поиска книги по автору происходит следующим образом:

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

Процесс записи книги на человека:

* библиотекарь вручную вводит ФИО человека, взявшего книгу.

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 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.

Чтобы добавить новую книгу, выполните следующие шаги:

1. Введите 1 в основном меню для выбора опции «Добавить книгу»;
2. Введите название книги, автора, издательство и год издания;
3. Подтвердите создание книги нажав клавишу «Enter»;
4. На экране появится сообщение «Книга сохранена.».

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

Если вводимая пользователем информация не соответствует заявленным типам данных, то на экране будет показано сообщение «Введены некорректные данные. Попробуйте ещё раз.».

* 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 выполните следующие шаги:

1. Введите 2 в основном меню для выбора опции «Удалить книгу»;
2. Введите название книги;
3. Подтвердите удаление книги нажав клавишу «Enter»;
4. На экране появится сообщение «Книга успешно удалена.» или «Книга не найдена.».

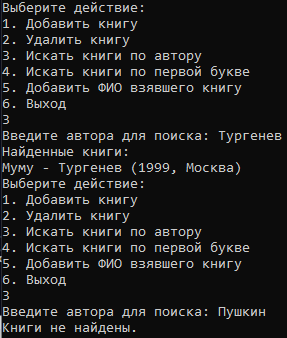
* Поиск книг:

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

Чтобы осуществить поиск книг по автору необходимо выполнить следующие шаги:

1. Введите 3 в главном меню для выбора опции «Искать книги по автору»;
2. Введите автора, которого хотите найти;
3. Подтвердите поиск книги нажав клавишу «Enter»;
4. Программа покажет все подходящие книги.

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



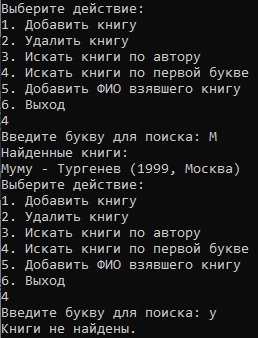
Чтобы осуществить поиск книг по первой букве необходимо выполнить следующие шаги:

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

Если книги на данную букву не существует, пользователь увидит сообщение «Книги не найдены.».

Все текстовые данные должны выводиться на исходном языке пользователя. Числовые данные выводятся арабскими цифрами.

Описать форматы возможных запросов поиска пользователем



* 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 выполнить следующие шаги:

1. Введите 5 в основном меню для выбора опции «Добавить ФИО взявшего книгу»;
2. Введите название книги;
3. Подтвердите название книги нажав клавишу «Enter»;
4. Введите ФИО человека, взявшего книгу;
5. Подтвердите ФИО нажав клавишу «Enter».

Если запрашиваемая книга отсутствует, то пользователь увидит сообщение «Книга не найдена».

Чтобы вернуть книгу необходимо выполнить следующие шаги:

1. Введите 6 в основном меню для выбора опции «Вернуть книгу»;
2. Введите название книги;
3. Подтвердите название книги нажав клавишу «Enter».

Если запрашиваемая книга отсутствует, то пользователь увидит сообщение «Книга не найдена».

* Выход из программы:

Чтобы выйти из программы необходимо выполнить следующие шаги:

1. Введите 7 в основном меню для выбора опции «Выход».

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

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

**Дорисуй что он хочет, я не шарю**

**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\_CSV\_файла Запись\_чтение\_CSV — экземпляр класса Запись\_CSV\_файла для работы с CSV файлом.

List<Записанная\_книга> книги — список всех книг в библиотеке.

const string Путь\_к\_файлу — путь к файлу, в котором хранятся данные о книгах.

Методы класса:

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

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

void Меню() — отображает меню выбора действий.

void Выполнить() — запускает цикл взаимодействия с пользователем.

void Поиск\_по\_автору()

void Поиск\_по\_букве()

void Добавление\_записи()

void Возврат\_записи()

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:** Книга

Данные класса:

string Название— название книги.

string Автор — имя автора книги.

string Издательство — издательство, выпустившее книгу.

int Год\_издания — год издания книги.

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

Теперь все выписанные книги наследуют класс Книги

Методы класса:

string 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. Этот класс необходим, так как он является родителем класса Записанная\_книга, используемого для всех операций с книгами в библиотеке.

**Класс:** Записанная\_книга

Данные класса:

string Записано\_за — ФИО взявшего книгу.

Методы класса:

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

**Описание:**

Класс Записанная\_книга является наследником класса Книга. Каждый объект этого класса содержит информацию класса Книга 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

Данные класса:

string Путь\_к\_файлу — путь к CSV файлу.

Методы класса:

Void Сохранить\_в\_CSV(List<Записанная\_книга>книги) — метод для сохранения списка книг в CSV файл.

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.

CSV\_файл представляет из себя таблицу Mc Excel и предназначен для долговременного хранения данных о книгах. При зрительном чтении CSV\_файла информация, хранимая в нём может отображаться не в соответствии с кодировкой. Данная ошибка может возникать из-за индивидуальных настроек кодировки Mc Excel. Также на работу программы данная ошибка не влияет и не является критической.

Форма записи книги без указания взятия книги человеком:



Форма записи книги с указанием взятия книги человеком:



Пример ошибки:



Данные класса:

Путь\_к\_файлу (const string) — путь к CSV-файлу, где хранятся записи о книгах, который не должен изменяться.

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

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

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

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

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

**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 константным (const), так как оно определяет местоположение CSV-файла, которое не меняется во время исполнения программы.

**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.



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.

Чтобы избежать потери данных или несоответствий при неожиданном закрытии программы пользователем, каждый раз, когда методу требуется список объектов Книга, нужно обращаться к методу Прочитать\_из\_CSV() для обновления данных. Это гарантирует, что программа будет использовать актуальные данные даже после изменения файла CSV вне программы.

1. **Фильтрация книг по автору (метод Where)**: Where применяется для фильтрации списка книг, оставляя только те, авторы которых совпадают с введенным именем.
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#

Изменим после полной готовности программы

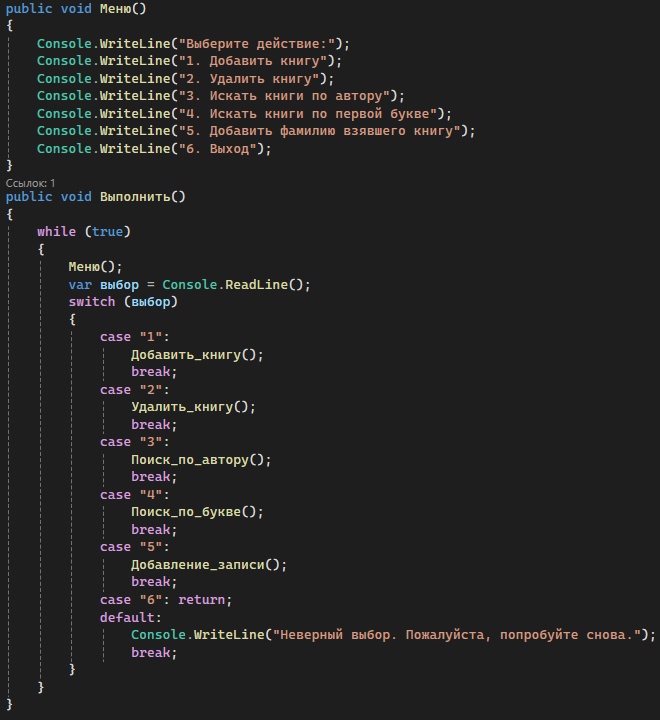


Рисунок 1 – Метод меню для выбора действия

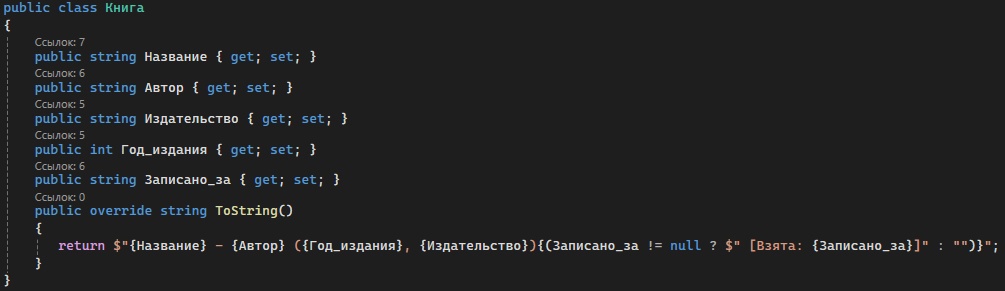


Рисунок 2 – Определение класса "Книга" с определением свойств и переопределением метода ToString

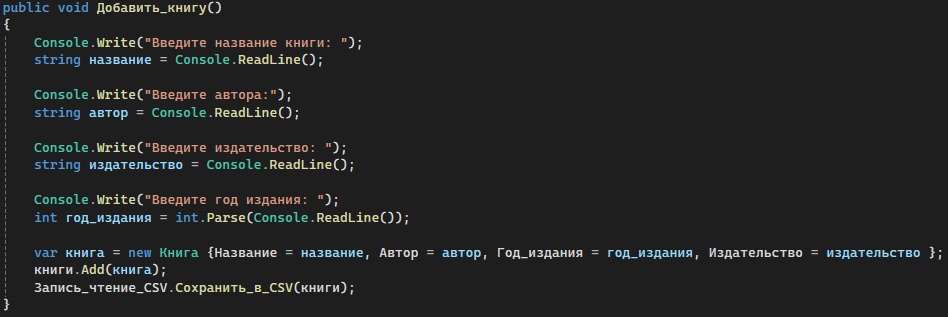


Рисунок 3 - Метод добавления новой книги

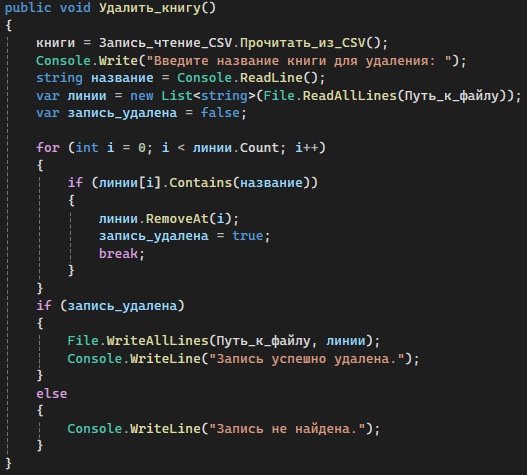


Рисунок 4 – Метод удаления книги

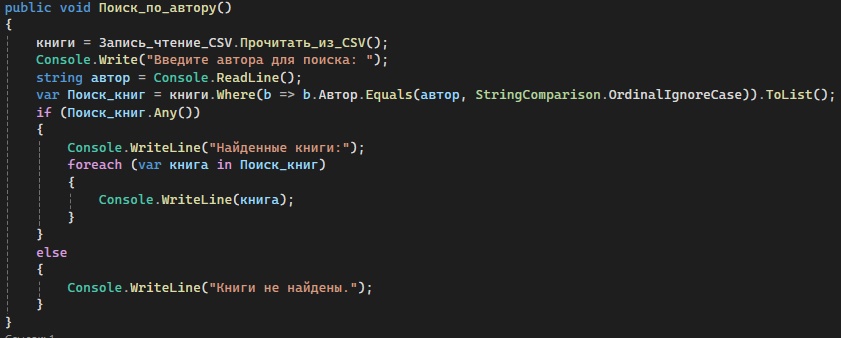


Рисунок 5 – Метод поиска книг по автору



Рисунок 6 – Метод поиска книги по букве

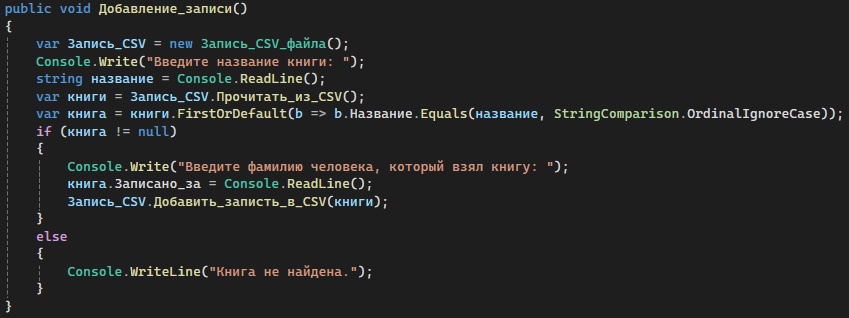


Рисунок 7 - Метод добавления отметки взятия книги человеком

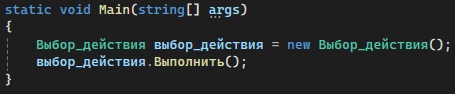


Рисунок 8 – Точка входа в программу

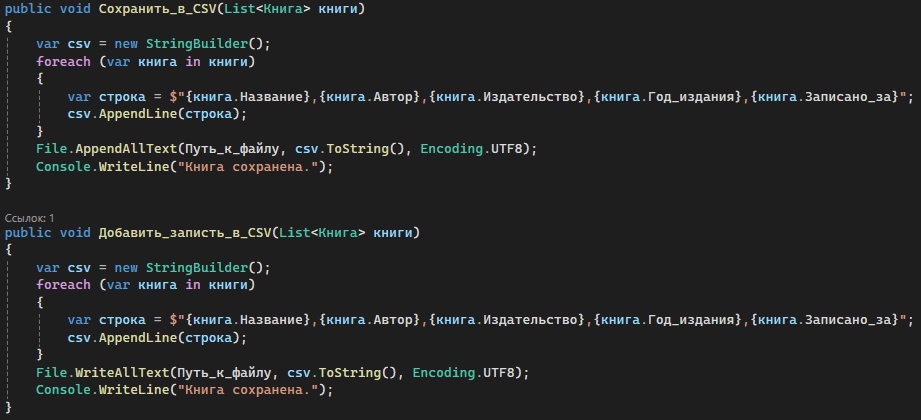


Рисунок 9 - Методы записи информации в CSV файл

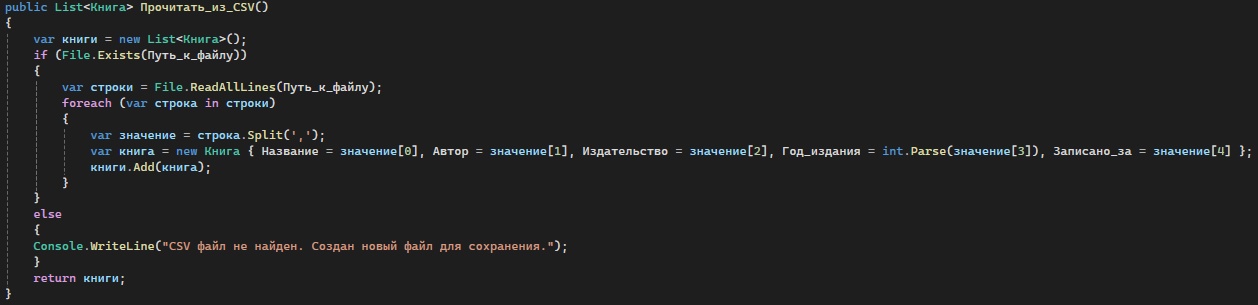


Рисунок 10 - Метод чтения информации из CSV файла