Лабораторная работа №6
Чтение и запись текстовых файлов. StreamReader и StreamWriter

Краткие теоретические сведения

1. Структуры

Кроме базовых элементарных типов данных и перечислений в C# имеется и составной тип данных, который называется структурой. Структуры могут содержать в себе обычные переменные и методы.

Для примера создадим структуру Book, в которой будут храниться переменные для названия, автора и года издания книги. Кроме того, структура будет содержать метод для вывода информации о книге на консоль:

```
    struct Book
    {
        public string name;
        public string author;
        public int year;

        public void Info()
        {
            Console.WriteLine($"Книга '{name}' (автор {author}) была издана в {year}
году");
        }
    }
```

Чтобы можно было использовать переменные и методы структуры из любого места программы мы ставим перед переменными и методом модификатор доступа public

Используем структуру на практике:

```
class Program
{
    static void Main(string[] args)
    {
        Book book;
        book.name = "Война и мир";
        book.author = "Л. Н. Толстой";
        book.year = 1869;

        //Выведем информацию о книге book на экран
        book.Info();

        Console.ReadLine();
    }
}
```

Структуру можно задать как внутри пространства имен (как в данном случае), так и внутри класса, но не внутри метода.

По сути, структура Book представляет новый тип данных. Мы также можем использовать массив структур:

```
Book[] books = new Book[3];
books[0].name = "Война и мир";
books[0].author = "Л. Н. Толстой";
books[0].year = 1869;

books[1].name = "Преступление и наказание";
books[1].author = "Ф. М. Достоевский";
books[1].year = 1866;

books[2].name = "Отцы и дети";
books[2].author = "И. С. Тургенев";
books[2].year = 1862;

foreach (Book b in books)
{
    b.Info();
}
```

Класс FileStream не очень удобно применять для работы с текстовыми файлами. К тому же для этого в пространстве System.IO определены специальные классы: StreamReader и StreamWriter.

2. Строки

2.1 Строки и класс System.String

Довольно большое количество задач, которые могут встретиться при разработке приложений, так или иначе связано с обработкой строк – парсинг веб-страниц, поиск в тексте, какие-то аналитические задачи, связанные с извлечением нужной информации из текста и т.д. Поэтому в этом плане работе со строками уделяется особое внимание.

В языке C# строковые значения представляет тип string, а вся функциональность работы с данным типом сосредоточена в классе System.String. Собственно string является псевдонимом для класса System.String. Объекты этого класса представляют текст как последовательность символов Unicode. Максимальный размер объекта String может составлять в памяти 2 ГБ, или около 1 миллиарда символов.

2.2 Создание строк

Создавать сроки можно, как используя переменную типа string и присваивая ей значение, так и применяя один из конструкторов класса String:

```
string s1 = "hello";
string s2 = null;

string s3 = new String('a', 6); // результатом будет строка "аааааа"
string s4 = new String(new char[] { 'w', 'o', 'r', 'l', 'd' });
```

Конструктор String имеет различное число версий. Так, вызов конструктора new String('a', 6) создаст строку "аааааа". И так как строка представляет ссылочный тип, то может хранить значение null.

2.3 Строка как набор символов

Так как строка хранит коллекцию символов, в ней определен индексатор для доступа к этим символам. Применяя индексатор, мы можем обратиться к строке как к массиву символов и получить по индексу любой из ее символов:

```
string s1 = "hello";
char ch1 = s1[1]; // символ 'e'
Console.WriteLine(ch1);
Console.WriteLine(s1.Length);
```

Используя свойство Length, как и в обычном массиве, можно получить длину строки.

2.4 Основные методы строк

Основная функциональность класса String раскрывается через его методы, среди которых можно выделить следующие:

- **Compare**: сравнивает две строки с учетом текущей культуры (локали) пользователя
- **CompareOrdinal**: сравнивает две строки без учета локали
- **Contains**: определяет, содержится ли подстрока в строке
- **Concat**: соединяет строки
- **CopyTo**: копирует часть строки или всю строку в другую строку
- **EndsWith**: определяет, совпадает ли конец строки с подстрокой
- **Format**: форматирует строку
- **IndexOf**: находит индекс первого вхождения символа или подстроки в строке
- **Insert**: вставляет в строку подстроку
- **Join**: соединяет элементы массива строк

- **LastIndexOf**: находит индекс последнего вхождения символа или подстроки в строке
- **Replace**: замещает в строке символ или подстроку другим символом или подстрокой
- **Split**: разделяет одну строку на массив строк
- **Substring**: извлекает из строки подстроку, начиная с указанной позиции
- **ToLower**: переводит все символы строки в нижний регистр
- **ToUpper**: переводит все символы строки в верхний регистр
- **Trim**: удаляет начальные и конечные пробелы из строки

2.5 Операции со строками

2.5.1 Конкатенация

Конкатенация строк или объединение может производиться как с помощью операции +, так и с помощью метода Concat:

```
string s1 = "hello";
string s2 = "world";
string s3 = s1 + " " + s2; // результат: строка "hello world"
string s4 = String.Concat(s3, "!!!"); // результат: строка "hello world!!!"

Console.WriteLine(s4);
```

Метод Concat является статическим методом класса String, принимающим в качестве параметров две строки. Также имеются другие версии метода, принимающие другое количество параметров.

Для объединения строк также может использоваться метод Join:

```
string s5 = "apple";
string s6 = "a day";
string s7 = "keeps";
string s8 = "a doctor";
string s9 = "away";
string[] values = new string[] { s5, s6, s7, s8, s9 };

String s10 = String.Join(" ", values);
// результат: строка "apple a day keeps a doctor away"
```

Метод Join также является статическим. Использованная выше версия метода получает два параметра: строку-разделитель (в данном случае пробел) и массив строк, которые будут соединяться и разделяться разделителем.

2.5.2 Сравнение строк

Для сравнения строк применяется статический метод Compare:

```
string s1 = "hello";
string s2 = "world";

int result = String.Compare(s1, s2);
if (result < 0)
{
    Console.WriteLine("Строка s1 перед строкой s2");
}
else if (result > 0)
{
    Console.WriteLine("Строка s1 стоит после строки s2");
}
else
{
    Console.WriteLine("Строки s1 и s2 идентичны");
}
// результатом будет "Строка s1 перед строкой s2"
```

Данная версия метода Compare принимает две строки и возвращает число. Если первая строка по алфавиту стоит выше второй, то возвращается число меньше нуля. В противном случае возвращается число больше нуля. И третий случай - если строки равны, то возвращается число 0.

В данном случае так как символ h по алфавиту стоит выше символа w, то и первая строка будет стоять выше.

2.5.3 Поиск в строке

С помощью метода IndexOf мы можем определить индекс первого вхождения отдельного символа или подстроки в строке:

```
string s1 = "hello world";
char ch = 'o';
int indexOfChar = s1.IndexOf(ch); // равно 4
Console.WriteLine(indexOfChar);

string subString = "wor";
int indexOfSubstring = s1.IndexOf(subString); // равно 6
Console.WriteLine(indexOfSubstring);
```

Подобным образом действует метод LastIndexOf, только находит индекс последнего вхождения символа или подстроки в строку.

Еще одна группа методов позволяет узнать начинается или заканчивается ли строка на определенную подстроку. Для этого предназначены методы StartsWith и EndsWith. Например, у нас есть задача удалить из папки все файлы с расширением exe:

```
string path = @"C:\SomeDir";

string[] files = Directory.GetFiles(path);

for (int i = 0; i < files.Length; i++)
```

```
{
    if (files[i].EndsWith(".exe"))
        File.Delete(files[i]);
}
```

2.5.4 Разделение строк

С помощью функции Split мы можем разделить строку на массив подстрок. В качестве параметра функция Split принимает массив символов или строк, которые и будут служить разделителями. Например, подсчитаем количество слов в сроке, разделив ее по пробельным символам:

```
string text = "И поэтому все так произошло";

string[] words = text.Split(new char[] { ' ' });

foreach (string s in words)
{
    Console.WriteLine(s);
}
```

2.5.5 Поиск в строке

С помощью метода **IndexOf** мы можем определить индекс первого вхождения отдельного символа или подстроки в строке:

```
string s1 = "hello world";
char ch = 'o';
int indexOfChar = s1.IndexOf(ch); // равно 4
Console.WriteLine(indexOfChar);

string subString = "wor";
int indexOfSubstring = s1.IndexOf(subString); // равно 6
Console.WriteLine(indexOfSubstring);
```

Подобным образом действует метод **LastIndexOf**, только находит индекс последнего вхождения символа или подстроки в строку.

2.5.6 Разделение строк

С помощью функции **Split** мы можем разделить строку на массив подстрок. В качестве параметра функция Split принимает массив символов или строк, которые и будут служить разделителями. Например, подсчитаем количество слов в сроке, разделив ее по пробельным символам:

```
string text = "И поэтому все так произошло";

string[] words = text.Split(' ');

foreach (string s in words)
{
```

```
    Console.WriteLine(s);
}
```

2.5.7 Замена подстроки

Чтобы заменить один символ или подстроку на другую, применяется метод **Replace**:

```
string text = "хороший день";

text = text.Replace("хороший", "плохой");
Console.WriteLine(text);

text = text.Replace("о", "");
Console.WriteLine(text);
```

Во втором случае применения функции Replace строка из одного символа "о" заменяется на пустую строку, то есть фактически удаляется из текста. Подобным способом легко удалять какой-то определенный текст в строках.

2.5.8 Вставка

Для вставки одной строки в другую применяется функция **Insert**:

```
string text = "Хороший день";
string subString = "замечательный ";

text = text.Insert(8, subString);
Console.WriteLine(text);
```

Первым параметром в функции Insert является индекс, по которому надо вставлять подстроку, а второй параметр - собственно подстрока.

2.5.9 Удаление строк

Удалить часть строки помогает метод **Remove**:

```
string text = "Хороший день";
// индекс последнего символа
int ind = text.Length - 1;
// вырезаем последний символ
text = text.Remove(ind);
Console.WriteLine(text);

// вырезаем первые два символа
text = text.Remove(0, 2);
```

Первая версия метода Remove принимает индекс в строке, начиная с которого надо удалить все символы. Вторая версия принимает еще один параметр - сколько символов надо удалить.

3. Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла. Среди его методов можно выделить следующие:

Close: закрывает считываемый файл и освобождает все ресурсы

Peek: возвращает следующий доступный символ, если символов больше нет, то возвращает -1

Read: считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов

ReadLine: считывает одну строку в файле

ReadToEnd: считывает весь текст из файла

Считаем текст из файла различными способами:

```
string path = @"C:\SomeDir\hta.txt";

try
{
    Console.WriteLine("******считываем весь файл********");
    StreamReader sr1 = new StreamReader(path);
    Console.WriteLine(sr1.ReadToEnd());

    Console.WriteLine();
    Console.WriteLine("******считываем построчно********");

    StreamReader sr2 = new StreamReader(path, System.Text.Encoding.Default);
    string line;
    while ((line = sr2.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }

    Console.WriteLine();
    Console.WriteLine("******считываем блоками********");

    StreamReader sr3 = new StreamReader(path, System.Text.Encoding.Default);
    char[] array = new char[4];
    // считываем 4 символа
    sr3.Read(array, 0, 4);
    Console.WriteLine(array);
}
catch (Exception e)
{
     Console.WriteLine(e.Message);
}
```

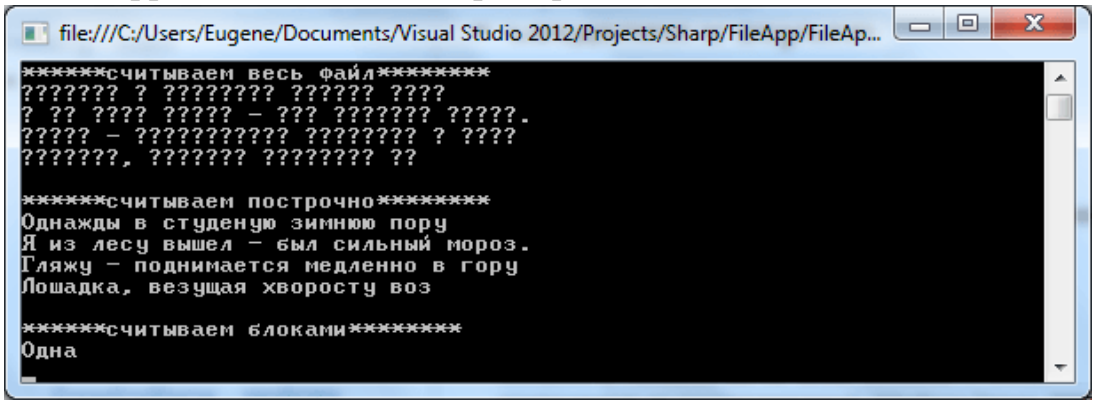
Как и в случае с классом FileStream здесь используется конструкция using.

В первом случае мы разом считываем весь текст с помощью метода ReadToEnd().

Во втором случае считываем построчно через цикл while: while ((line = sr.ReadLine()) != null) - сначала присваиваем переменной line результат функции sr.ReadLine(), а затем проверяем, не равна ли она null. Когда объект sr дойдет до конца файла и больше строк не останется, то метод sr.ReadLine() будет возвращать null.

В третьем случае считываем в массив четыре символа.

Обратите внимание, что в последних двух случаях в конструкторе StreamReader указывалась кодировка System.Text.Encoding.Default. Свойство Default класса Encoding получает кодировку для текущей кодовой страницы ANSI. Также через другие свойства мы можем указать другие кодировки. Если кодировка не указана, то при чтении используется UTF8. Иногда важно указывать кодировку, так как она может отличаться от UTF8, и тогда мы получим некорректный вывод. Например,



4. Запись в файл и StreamWriter

Для записи в текстовый файл используется класс StreamWriter. Свою функциональность он реализует через следующие методы:

Close: закрывает записываемый файл и освобождает все ресурсы

Flush: записывает в файл оставшиеся в буфере данные и очищает буфер.

Write: записывает в файл данные простейших типов, как int, double, char, string и т.д.

WriteLine: также записывает данные, только после записи добавляет в файл символ окончания строки

Рассмотрим запись в файл на примере:

```
string readPath = @"C:\SomeDir\hta.txt";
string writePath = @"C:\SomeDir\ath.txt";

string text = "";
try
{
    StreamReader sr = new StreamReader(readPath, System.Text.Encoding.Default);
    text = sr.ReadToEnd();

    StreamWriter sw = new StreamWriter(writePath, false, System.Text.Encoding.Default)
    sw.WriteLine(text);
    sw.Close();

    StreamWriter sw = new StreamWriter(writePath, true, System.Text.Encoding.Default);
    sw.WriteLine("Дозапись");
    sw.Write(4.5);
    sw.Close();

}
catch (Exception e)
{
    Console.WriteLine(e.Message);
}
```

Здесь сначала мы считываем файл в переменную text, а затем записываем эту переменную в файл, а затем через объект StreamWriter записываем в новый файл.

Класс StreamWriter имеет несколько конструкторов. Здесь мы использовали один из них: new StreamWriter(writePath, false, System.Text.Encoding.Default). В качестве первого параметра передается путь к записываемому файлу. Второй параметр представляет булевую переменную, которая определяет, будет файл дозаписываться или перезаписываться. Если этот параметр равен true, то новые данные добавляются в конце к уже имеющимся данным. Если false, то файл перезаписывается. И если в первом случае файл перезаписывается, то во втором делается дозапись в конец файла.

Третий параметр указывает кодировку, в которой записывается файл.

Задания на лабораторную работу:

Задание: создать текстовый файл с произвольной информацией. Организовать просмотр содержимого файла. Организовать чтение и обработку данных из файла в соответствии с индивидуальным заданием. Сохранить полученные результаты в новый текстовый файл.

№ вар.	Задача
1	«Человек»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения о самом молодом человеке.
2	«Школьник»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); школа; класс. Вывести сведения про всех учеников пятых классов.
3	«Студент»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); ВУЗ; курс; группа; средний бал; специальность. Вывести сведения о студентах у которых средний балл ниже 70 баллов.
4	«Покупатель»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер кредитной карточки; банковского счета. Вывести данные о покупателях с города Муром.
5	«Пациент»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); номер больницы; отделение; номер медицинской карты; диагноз; группа крови. Вывести данные про пациентов с 18 отделения.
6	«Владелец автомобиля»: фамилия; имя; отчество; номер телефона; домашний адрес (почтовый

	индекс, страна, область, район, город, улица, дом, квартира) марка автомобиля; номер автомобиля; номер техпаспорта. Вывести данные про автомобили марки ”Ваз”.
7	«Военнослужащий»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); должность; звание. Вывести данные про военнослужащих в звании “лейтенант”.
8	«Рабочий»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); национальность; дата рождения (год, месяц число); № цеха; табельный номер; образование; год поступления на работу. Вывести данные про рабочих, поступивших на работу в 2010 году.
9	«Владелец телефона»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); № телефона. Вывести данные про владельцев телефона номер, которого начинается на 720.
10	«Абитуриент»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); оценки по экзаменам; проходной балл. Вывести данные про абитуриентов, проходной балл которых равен больше 4 .
11	«Государство»: название страны; столица; государственный язык; население; площадь территории; денежная единица; государственный строй; глава государства. Вывести данные про государства, население которых больше 20 млн жителей.
12	«Автомобиль»: марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена. Вывести данные про автомобили, которым больше 2 лет.
13	«Товар»: наименование; стоимость; срок хранения; сорт; дата выпуска; срок годности. Вывести данные про товары срок годности которых истекает в этом году.
14	«Кинолента»: название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль.

	Вывести данные про фильмы режиссера Ежи Гофмана.
15	«Рейс»: марка автомобиля; номер автомобиля; пункт назначения; грузоподъемность (в тоннах); стоимость единицы груза; общая стоимость груза. Вывести данные про автомобили, грузоподъемность которых больше 2 тонн.
16	«Книга»: название; автор (фамилия; имя); год выхода; издательство; себестоимость; цена; прибыль. Вывести данные про книги авторов, фамилия которых начинается с буквы “К”.
17	«Здание»: адрес; тип здания; количество этажей; количество квартир; срок эксплуатации; срок до капитального ремонта (25 лет - срок эксплуатации). Вывести данные про здания срок эксплуатации, которых больше 50 лет.
18	«Программист»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); образование; номер телефона. Вывести сведения о программистах, которым меньше 25 лет.
19	«Ученый»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); ученая степень, должность, номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения про ученых кандидатов технических наук.
20	«Пенсионер»: фамилия; имя; отчество; пол; национальность; дата рождения (год, месяц число); номер телефона; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести сведения про всех пенсионеров, которые на пенсии больше 5 лет.
21	«Футболист»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; название команды; номер в команде; амплуа; результативность (количество голов); количество игр. Вывести сведения про футболистов, которые провели за свою команду больше 50 матчей.
22	«Манекенщица»: фамилия; имя; отчество; пол; национальность; рост; вес; дата

	рождения (год, месяц число); домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира). Вывести данные про самую молодую манекенщицу.
23	«Международная компания»: название; интернет сайт; адрес главного офиса (почтовый индекс, страна, область, район, город, улица, дом, квартира) продолжительность пребывания на мировом рынке; количество сотрудников; количество филиалов в Европе. Вывести международные компании, количество сотрудников у которых больше 10000.
24	«Телохранитель»: фамилия; имя; отчество; домашний адрес (почтовый индекс, страна, область, район, город, улица, дом, квартира); дата рождения (год, месяц число). Вывести данные про старшего телохранителя”.
25	«Зоопарк»: Название животного; количество вида; адрес зоопарка (почтовый индекс, страна, область, район, город, улица, дом, квартира); общее количество животных, количество работников. Вывести сведения про зоопарки, в которых есть уссурийские тигры.
26	«Программное обеспечение»: название; название компании производителя; год выхода; цена. Вывести данные про программное обеспечение, которое дороже 2000 рублей.
27	«Мультфильм»: название; режиссер (фамилия; имя); год выхода; страна; стоимость; доход; прибыль. Вывести данные про мультфильмы компании “Walt Disney”.
28	«Баскетболист»: фамилия; имя; отчество; пол; национальность; рост; вес; дата рождения (год, месяц число); номер телефона; название команды; номер в команде; амплуа; результативность (количество очков); количество игр. Вывести сведения про баскетболистов, которых забросили за свою команду больше 150 очков.
29	«Область»: название области; областной центр; население; площадь территории; губернатор. Вывести данные про области, население которых меньше 2 млн. жителей.
30	«Мотоцикл»: марка; цвет; серийный номер; регистрационный номер; год выпуска; год техосмотра; цена. Вывести данные про мотоциклы марки ”Harley Davidson”.

Пример

Students

```
5
Иванов Иван Иванович
13.04.2001
ПИНз-120
4,6
Петров Сергей Владимирович
15.08.2000
ПИНз-119
4,0
Васильев Иван Бориович
29.03.2000
ПИНз-119
3,0
Курышев Олег Егорович
03.06.2001
ПИНз-120
4,9
Васечкин Петр Денисович
01.03.2001
ПИНз-119
5,0
```

```
using System;
using System.IO;

namespace ConsoleApp4
{
    struct Student
    {
        public string FIO;
        public DateTime Birthday;
        public string Group;
        public double AvgMark;

        public override string ToString()
        {
            return $"ФИО: {FIO}, Группа: {Group}, Ср. балл: {AvgMark}, дата рождения:
{Birthday.ToShortDateString()}";
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            //Путь к исходному файлу
            string path = @"C:\Intel\students.txt";

            if(!File.Exists(path))
            {
                Console.WriteLine($"Файл {path} не существует! Дальнейшая работа не
возможна...");
                Console.ReadKey();
                return;
            }

            int ItemsCount;
            Student[] a;

            //В блоке using выполняем безопасное открытие файла
            using (StreamReader sr = new StreamReader(path, System.Text.Encoding.Default))
```

```
            {
                //считываем первую строку - в ней записано число объектов, записанных в файле
                string ItemsCountStr = sr.ReadLine();

                //функция TryParse пытается преобразовать строку в число.
                //Возвращает истину если удалось, ложь - иначе.
                //Само преобразованное значение возвращает через выходной параметр,
                //помеченный словом out
                if (!int.TryParse(ItemsCountStr, out ItemsCount))
                {
                    Console.WriteLine("В первой строке файла нет числа объектов. Дальнейшая
работа невозможна.");
                    return;
                }

                //Создаем массив для хранения данных
                a = new Student[ItemsCount];
                //В цикле считываем даные
                for (int i = 0; i < ItemsCount; i++)
                {
                    //Построчно считываем данные
                    a[i].FIO = sr.ReadLine();
                    a[i].Birthday = DateTime.Parse(sr.ReadLine());
                    a[i].Group = sr.ReadLine();
                    a[i].AvgMark = double.Parse(sr.ReadLine());
                    Console.WriteLine(a[i]);
                }
            }

            Console.WriteLine($"Считано {ItemsCount} объектов.");

            //Задание 1. Вывести на консоль
            //и записать в файл студентов со средним баллом больше 4,5
            string writePath = @"C:\Intel\Students45.txt";

            Console.WriteLine("Студенты со средним баллом больше 4,5");

            //Открываем файл
            StreamWriter sw = new StreamWriter(writePath, false,
System.Text.Encoding.Default);

            for (int i = 0; i < ItemsCount; i++)
            {
                //Если средний балл i-го студента больше 4.5
                if (a[i].AvgMark > 4.5)
                {
                    Console.WriteLine(a[i].ToString());
                    //Записываем его в файл
                    sw. WriteLine(a[i].ToString());
                }
            }
            //НЕ ЗАБЫВАЕМ ЗАКРЫТЬ ФАЙЛ, ЕСЛИ ЕГО ОТКРЫВАЛИ НЕ В БЛОКЕ using
            sw.Close();

            //Задание 2. Вывести студентов, фамилия которых начинается с введенной подстроки
            string writePath2 = @"C:\Intel\StudentsFioStarts.txt";

            Console.Write("Введите подстроку для поиска: ");
            string FioStarts = Console.ReadLine();

            //Открываем файл
            StreamWriter sw2 = new StreamWriter(writePath2, false,
System.Text.Encoding.Default);

            for (int i = 0; i < ItemsCount; i++)
            {
                //Если ФИО студента начинается с введенной фразы
                if (a[i].FIO.StartsWith(FioStarts))
                {
```

```
                Console.WriteLine(a[i].ToString());
                //Записываем его в файл
                sw2. WriteLine(a[i].ToString());
            }
        }

        //НЕ ЗАБЫВАЕМ ЗАКРЫТЬ ФАЙЛ, ЕСЛИ ЕГО ОТКРЫВАЛИ НЕ В БЛОКЕ using
        sw2.Close();

        Console.WriteLine("Работа окончена. Нажмите любую клавишу для выхода...");
        Console.ReadKey();

    }
}
}
```

Результат работы программы