СОЦИАЛЬНО-ГУМАНИТАРНЫЙ КОЛЛЕДЖ
УО «МОГИЛЕВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ А.А. КУЛЕШОВА»

Дисциплина
«Конструирование программ и языки программирования»

Работа с файлами в C#

(4 часа)

Методические рекомендации к лабораторной работе №15

Могилев 2018

Понятия «Файл», «Функции работы с файлами». Методические указания по лабораторной работе №15 «Конструирование программ и языки программирования». Для учащихся 3 курса очной формы обучения специальности 2–40 01 01 «Программное обеспечение информационных технологий.

Оглавление

1 Цель работы 4
2 Ход работы 5
3 Краткие теоретические сведения 6
3.1 Как создать файл? 6
3.2 Как удалить файл? 7
3.3 Чтение из файла 8
3.4 Запись в файл 9
3.5 Как создать папку? 10
3.6 Как удалить папку? 10
3 Задания 13
4 Контрольные вопросы 15

1 Цель работы

Целью лабораторной работы является:

1. реализовать методы, позволяющие сохранять и читать данные объектов класса-коллекции из текстового файла;
2. применить при чтении/записи потоки, поддерживающие сжатие информации.

2 Ход работы

1. Изучение теоретического материала.
2. Выполнение практических индивидуальных заданий по вариантам (вариант уточняйте у преподавателя).
3. Оформление отчета.
   3.1. Отчет оформляется индивидуально каждым студентом. Отчет должен содержать задание, алгоритм и листинг программы.
   3.2. Отчет по лабораторной работе выполняется на листах формата А4. В состав отчета входят:
   1) титульный лист;
   2) цель работы;
   3) текст индивидуального задания;
   4) выполнение индивидуального задания.

3. 4. Контрольные вопросы.

3 Краткие теоретические сведения

Файл – это набор данных, который хранится на внешнем запоминающем устройстве (например на жестком диске). Файл имеет имя и расширение. Расширение позволяет идентифицировать, какие данные и в каком формате хранятся в файле.

Под работой с файлами подразумевается:

- создание файлов;
- удаление файлов;
- чтение данных;
- запись данных;
- изменение параметров файла (имя, расширение…);
- другое.

В **C#** есть пространство имен `System.IO`, в котором реализованы все необходимые нам классы для работы с файлами. Чтобы подключить это пространство имен, необходимо в самом начале программы добавить строку `using System.IO`. Для использования кодировок еще добавим пространство `using System.Text`;

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
```

3.1 Как создать файл?

Для создания пустого файла, в классе `File` есть метод `Create()`. Он принимает один аргумент – путь. Ниже приведен пример создания пустого текстового файла `new_file.txt` на диске D:

```
static void Main(string[] args)
        {
            File.Create("D:\\new_file.txt");
        }
```

Если файл с таким именем уже существует, он будет переписан на новый пустой файл.

Метод `WriteAllText()` создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле:

```
static void Main(string[] args)
        {
            File.WriteAllText("D:\\new_file.txt", "текст");
        }
```

Метод `AppendAllText()` работает, как и метод `WriteAllText()` за исключением того, что новый текст дописывается в конец файла, а не переписывает всё что было в

файле:

```
static void Main(string[] args)
        {
            File.AppendAllText("D:\\new_file.txt",    "текст
метода AppendAllTExt ()");
        }
```

3.2 Как удалить файл?

Метод **`Delete()`** удаляет файл по указанному пути:

```
static void Main(string[] args)
    {
      File.Delete("d:\\test.txt"); // удаление файла
    }
```

Кроме того, чтобы читать/записывать данные в файл с **C#** можно использовать потоки.

Поток – это абстрактное представление данных (в байтах), которое облегчает работу с ними. В качестве источника данных может быть файл, устройство ввода-вывода, принтер.

Класс **`Stream`** является абстрактным базовым классом для всех потоковых классов в **C#**. Для работы с файлами нам понадобится класс **`FileStream`** (файловый поток).

`FileStream` - представляет поток, который позволяет выполнять операции чтения/записи в файл.

```
static void Main(string[] args)
        {
          FileStream file = new FileStream("d:\\test.txt",
FileMode.Open, FileAccess.Read);//открывает файл только на
чтение
        }
```

Режимы открытия `FileMode`:

- `Append` – открывает файл (если существует) и переводит указатель в конец файла (данные будут дописываться в конец), или создает новый файл. Данный режим возможен только при режиме доступа FileAccess.Write;
- `Create` – создает новый файл(если существует – заменяет);
- `CreateNew` – создает новый файл (если существует – генерируется исключение);
- `Open` – открывает файл (если не существует – генерируется исключение);
- `OpenOrCreate` – открывает файл, либо создает новый, если его не существует;
- `Truncate` – открывает файл, но все данные внутри файла затирает (если файла не существует – генерируется исключение).

```
static void Main(string[] args)
        {
```

```
            FileStream file1 = new FileStream("d:\\test.txt",
FileMode.CreateNew); // создание нового файла
            FileStream file2 = new FileStream("d:\\test.txt",
FileMode.Open);// открытие существующего файла
            FileStream file3 = new FileStream("d:\\test.txt",
FileMode.Append); // открытие на дозапись в конец файла
        }
```

Режим доступа `FileAccess`:

- `Read` – открытие файла только на чтение. При попытке записи генерируется исключение;
- `Write` – открытие файла только на запись. При попытке чтения генерируется исключение;
- `ReadWrite` – открытие файла на чтение и запись.

3.3 Чтение из файла

Для чтения данных из потока нам понадобится класс `StreamReader`. В нем реализовано множество методов для удобного считывания данных. Ниже приведена программа, которая выводит содержимое файла на экран:

```
static void Main(string[] args)
        {
          FileStream file1 = new FileStream("d:\\test.txt",
FileMode.Open);// создаем файловый поток
            StreamReader reader = new StreamReader(file1);
// создаем "потоковый читатель" и связываем его с файловым
потоком
            Console.WriteLine(reader.ReadToEnd());
//считываем все данные с потока и выводим на экран
            reader.Close(); // закрываем поток
            Console.ReadLine();
        }
```

Метод `ReadToEnd()` считывает все данные из файла. `ReadLine()` – считывает одну строку (указатель потока при этом переходит на новую строку, и при следующем вызове метода будет считана следующая строка).

Свойство `EndOfStream` указывает, находится ли текущая позиция в потоке в конце потока (достигнут ли конец файла). Возвращает `true` или `false`.

Для того чтобы файл был корректно считан (без всяких непонятных символов), он должен быть сохранен в юникоде.

```
StreamReader streamReader = new StreamReader("name.txt");
            string str = "";
                while (!streamReader.EndOfStream)
                {
```

```
                str += streamReader.ReadLine();
            }
```

3.4 Запись в файл

Для записи данных в поток используется класс `StreamWriter`. Пример записи в файл:

```
static void Main(string[] args)
        {
          FileStream file1 = new FileStream("d:\\test.txt",
FileMode.C);// создаем файловый поток
            StreamWriter writer = new StreamWriter(file1);
// создаем "потоковый писатель" и связываем его с файловым
потоком
            writer.Write("текст"); //записываем в файл
            writer.Close(); // закрываем поток. Не закрыв
поток, в файл ничего не запишется
            Console.ReadLine();
        }
```

Метод `WriteLine()` записывает в файл построчно (то же самое, что и простая запись с помощью `Write()`, только в конце добавляется новая строка).

Нужно всегда помнить, что после работы с потоком, его нужно закрыть (освободить ресурсы), использовав метод `Close()`.

Кодировка, в которой будут считываться/записываться данные, указывается при создании `StreamReader/StreamWriter`:

```
static void Main(string[] args)
        {
          FileStream file1 = new FileStream("d:\\test.txt",
FileMode.C);
          StreamReader reader = new StreamReader(file1, En-
coding.Unicode);
          StreamWriter writer = new StreamWriter(file1, En-
coding.UTF8);
        }
```

Кроме того, при использовании `StreamReader` и `StreamWriter` можно не создавать отдельно файловый поток `FileStream`, а сделать это сразу при создании `StreamReader/StreamWriter`:

```
static void Main(string[] args)
   {
    StreamWriter writer = new StreamWriter("d:\\test.txt");
    writer.WriteLine("текст");
    writer.Close();
```

```
    }
```

3.5 Как создать папку?

С помощью статического метода `CreateDirectory()` класса `Directory`:

```
static void Main(string[] args)
        {
            Directory.CreateDirectory("d:\\ new_folder");
        }
```

3.6 Как удалить папку?

Для удаления папок используется метод `Delete()`:

```
static void Main(string[] args)
        {
            Directory.Delete("d:\\  new_folder");  // удале-
ние пустой папки
        }
```

Если папка не пустая, необходимо указать параметр рекурсивного удаления – `true`:

```
static void Main(string[] args)
        {
            Directory.Delete("d:\\  new_folder",  true);  //
удаление папки и всего, что внутри
        }
```

Практическит примерs

1. Данная программа считывает указанный пользователем файл построчно и выводит его на экран.

```
using System;
using System.IO;
using System.Text;

namespace ConsoleApplication
{
    //класс для чтения текстовых файлов
    class ReadSomeFile
    {
        static  void  Main(string[]  args  /*  параметра  командной
строки*/)
        {
            string FileName;
```

```
            //если  в командной строке параметров нет
            if (args.Length == 0)
            {
                Console.Write("Введите пусть к файлу: ");
                FileName = Console.ReadLine();
            }
            else
            {
                FileName = args[0];
            }

            try
            {
          //открываем поток для чтения с кодировкой по умолчанию
                StreamReader sr = new StreamReader(FileName, En-
coding.Default);
                string line;
                while ((line = sr.ReadLine ()) != null)
                {
                    //вывод на экран
                    Console.WriteLine(line);
                }
                sr.Close();
            }
            catch (Exception ex)
            {
                //Сообщение об оишбке
                Console.WriteLine(ex.Message);
            }
        }
    }
}
```

Данная программа записывает введенные пользователем с клавиатуры строки в файл, дописывая время начала и окончания работы пользователя.

```
//класс для работы с файлами
    class WriteSomeClass
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Введите любой текст!");
            Console.WriteLine("Ввод пусстой строки - окончание
ввода!");
            //открываем поток для запси в файл с кодировкой по
умолчанию
```

```
            StreamWriter sw = new StreamWriter("d:\\User.log",
true, Encoding.Default);
            string line;
            sw.WriteLine("---- Начало сеанса ----");
                // запись текущего времени
                sw.WriteLine(DateTime.Now);
            sw.WriteLine("------------------------");
            sw.WriteLine();

            do
            {
                // считываем строку с клавиатуры
                line = Console.ReadLine();
                // записываем строку в файл
                sw.WriteLine(line);
            } while (line != "");
            sw.WriteLine("---- Окончание сеанса ----");
            sw.WriteLine(DateTime.Now);
            sw.WriteLine("------------------------");
            //закрываем поток
            sw.Close();
        }
    }
```

3 Задания

1. Выполнить задание по вариантам:

Вариант 1

Написать программу, которая считывает из текстового файла три предложения и выводит их в обратном порядке.

Вариант 2

Написать программу, которая считывает текст из файла и выводит на экран только предложения, содержащие введенное с клавиатуры слово.

Вариант 3

Написать программу, которая считывает текст из файла и выводит на экран сначала вопросительные, а затем восклицательные предложения.

Вариант 4

Написать программу, которая считывает английский текст из файла и выводит на экран слова, начинающиеся с гласных букв.

Вариант 5

Написать программу, которая считывает текст из файла и выводит его на экран, меняя местами каждые два соседних слова.

Вариант 6

Написать программу, которая считывает текст из файла и выводит на экран только предложения, не содержащие запятых.

Вариант 7

Написать программу, которая считывает текст из файла и определяет, сколько в нем слов, состоящих не более чем из четырех букв.

Вариант 8

Написать программу, которая считывает текст из файла и выводит на экран только цитаты, то есть предложения, заключенные в кавычки.

Вариант 9

Написать программу, которая считывает текст из файла и выводит на экран только предложения, состоящие из заданного количества слов.

Вариант 10

Написать программу, которая считывает английский текст из файла и выводит на экран слова текста, начинающиеся и оканчивающиеся на гласные буквы.

Вариант 11

Написать программу, которая считывает текст из файла и выводит на экран только строки, не содержащие двузначных чисел.

Вариант 12

Написать программу, которая считывает текст из файла и выводит на экран только предложения, начинающиеся с тире, перед которым могут находиться только пробельные символы.

Вариант 13

Написать программу, которая считывает английский текст из файла и выводит его на экран, заменив прописной каждую первую букву слов, начинающихся гласной буквы.

Вариант 14

Написать программу, которая считывает текст из файла и выводит его на экран, заменив цифры от 0 до 9 словами «ноль», «один», «девять», начиная каждое предложение с новой строки.

Вариант 15

Написать программу, которая считывает текст из файла, находит самое длинное слово и определяет, сколько раз оно встретилось в тексте.

5 Контрольные вопросы

1. Что такое файл? Как его создать? Удалить? Извлечь данные?
2. Как создать и удалить папку?
3. Что такое кодировка? Зачем она нужна?