Лабораторная работа № 15. Работа с файлами.

Тема: Разработка программ создания и обработки файлов

Цель работы: Формирование умений и навыков использования файлов в программе и их обработки

Общие теоретические сведения:

Для работы с файлами, все необходимые (основные) классы находятся в пространстве имен **System.IO,** в таблице ниже (таблица 1) представлены эти классы.

Таблица 1. Классы пространства имени System.IO

Класс	*Применение*
Binary Reader и Writer	Чтение и запись простых типов данных
Directory, DirectoryInfo File, FileInfo	Создание, удаление и перемещение файлов и директорий. Получение подробной информации о файлах, при помощи свойств, определенных в этих классах.
FileStream	Доступ к файлам потоковым способом
MemoryStream	Доступ к данным хранящимся в памяти
StreamWriter и StreamReader	Чтение и запись текстовой информации
StringReader и StringWriter	Чтение и запись текстовой информации из строкового буфера

DirectoryInfo и FileInfo

Классы DirectoryInfo и FileInfo унаследованы от FileSystemInfo, который является запечатанными. Это значит, что Вы не можете унаследовать от него свой класс, но можете использовать свойства, определённые в нём. В таблице 2 перечислены его свойства и методы

Таблица 2. Свойства и методы классов DirectoryInfo и FileInfo

Свойства	*Назначение*
Attributes	Возвращает атрибуты файла в виде значений перечисления FileAttributes
CreationTime	Возвращает время создания файла
Exists	Проверяет является ли файл директорией или нет
Extension	Возвращает расширение файла
LastAccessTime	Возвращает время последнего доступа к файлу

FullName	Возвращает полный путь к файлу
LastWriteTime	Возвращает время последнего изменения файла
Name	Возвращает имя данного файла
Delete()	Удаляет файл. Будьте осторожны при использовании этого метода.

Класс **DirectoryInfo** содержит методы для создания, перемещение и удаление каталогов. Чтобы использовать вышеприведённые свойства, необходимо создать объект класса **DirectoryInfo** как показано в примере:

```
DirectoryInfo dir1 = new DirectoryInfo(@"F:\dirName");
```

После этого уже можно просмотреть свойства директории при помощи объекта dir1, как показано на фрагменте кода ниже:

```
Console.WriteLine("Full Name is : {0}", dir1.FullName);
Console.WriteLine("Attributes are : {0}",
                  dir1.Attributes.ToString());
```

Можно также использовать значения перечисления **FileAttributes**. Они приведены в таблице 3.

Таблица 3. Значения перечисления FileAttributes

Свойства	***Назначение***
Archive	Возвращает Архивный статус файла
Compressed	Позволяет узнать сжат файл или нет
Directory	Показывает является ли файл директорией или нет
Encrypted	Показывает закодирован файл или нет
Hidden	Показывает скрыт файл или нет
Offline	Показывает, что данные отсутствуют
ReadOnly	Показывает является ли файл только для чтения
System	Показывает, является ли файл системным (возможно файл в директории Windows)

Работа с файлами в директории

Предположим, вы хотите получить список всех файлов с расширением BMP в папке F:\Pictures. Для этого можно использовать следующий код:

```
DirectoryInfo dir = new DirectoryInfo(@"F:\dirName");
FileInfo[] bmpfiles = dir.GetFiles("*.bmp);
Console.WriteLine("Total number of bmp files", bmpfiles.Length);
Foreach( FileInfo f in bmpfiles)
{
  Console.WriteLine("Name is : {0}", f.Name);
  Console.WriteLine("Length of the file is : {0}", f.Length);
  Console.WriteLine("Creation time is : {0}", f.CreationTime);
  Console.WriteLine("Attributes of the file are : {0}",
                    f.Attributes.ToString());
}
```

Создание подкаталогов

Следующий фрагмент кода описывает как можно создать поддиректорию MySub в директории Sub:

```
DirectoryInfo dir = new DirectoryInfo(@"F:\dirName");
try
{
  dir.CreateSubdirectory("Sub");
  dir.CreateSubdirectory(@"Sub\MySub");
}
catch(IOException e)
{
  Console.WriteLine(e.Message);
}
```

Создание файлов при помощи класса FileInfo

Класс **FileInfo** позволяет создавать новые файлы, получать информацию, удалять и перемещать их. В этом классе также есть методы для открытия, чтения и записи в файл. В следующем примере показано, как можно создать текстовый файл и получить доступ к его информации (времени его создания, полное имя, и так далее):

```
FileInfo fi = new FileInfo(@"F:\Myprogram.txt");
FileStream fstr = fi.Create();
Console.WriteLine("Creation Time: {0}",f.CreationTime);
Console.WriteLine("Full Name: {0}",f.FullName);
Console.WriteLine("FileAttributes:
{0}",f.Attributes.ToString());

//Удаление файла Myprogram.txt.

Console.WriteLine("Press any key to delete the file");
Console.Read();
fstr.Close();
fi.Delete();
```

Описание метода Open()

В классе **FileInfo** есть метод под названием ***Open***(), с помощью которого можно создавать файлы, подставляя в параметры значения перечислений **FileMode** и **FileAccess**. Следующий фрагмент кода показывает, как это делается:

```
FileInfo f = new FileInfo("c:\myfile.txt");
FileStream s = f.Open(FileMode.OpenorWrite,
                      FileAccess.Read);
```

После этого, используя объект 's', можно читать и записывать в файл. В перегруженном методе ***Open***() можно только читать из файла. Для записи в файл необходимо в параметрах открытия использовать значение **FileAccess**.**ReadWrite**. Таблицы 4 и 5 содержат возможные значения **FileMode** и **FileAccess**.

Таблица 4. Значения перечисления FileMode

Значение	***Применение***
Append	Для открытия файла и добавления данных. Используется совместностно со значением FileAccess.Write.
Create	Для создания нового файла. Если файл уже существует, то он затирается.
CreateNew	Для создания нового файла. Если файл существует, то возникает исключение IOException.
Open	Для открытия существующего файла
OpenOrCreate	Для открытия существующего или создания нового файла. Если файл не существует, тобудет создан новый.
Truncate	Для урезания существующего файла

Таблица 5. Значения перечисления FileAccess

Значения	*Применение*
Read	Для чтения (получения) данных из файла
ReadWrite	Для записи в или чтения из файла
Write	Для записи данных в файл

Запись в текстовый файл при помощи класса StreamWriter

Текстовые данные или любую другую информацию можно записать в файл используя метод ***CreateText***() в классе **FileInfo**. Однако предварительно необходимо получить валидный **StreamWriter**. Именно **StreamWrite** обеспечивает необходимую функциональность для записи в файл. Следующий пример иллюстрирует это:

```
FileInfo f = new FileInfo("Mytext.txt")
StreamWriter w = f.CreateText();
w.WriteLine("This is from");
w.WriteLine("Chapter 6");
w.WriteLine("Of C# Module");
w.Write(w.NewLine);
w.WriteLine("Thanks for your time");
w.Close();
```

Чтение из текстового файла

Для чтения из текстового файла можно воспользоваться классом **StreamReader**. Для этого необходимо указать имя файла в статическом методе ***OpenText***() класса **File**. Следующий пример считывает содержимое файла, которое было записано в предыдущем примере:

```
Console.WriteLine("Reading the contents from the file");
StreamReader s = File.OpenText("Mytext.txt");
string read = null;
while ((read = s.ReadLine()) != null)
{
  Console.WriteLine(read);
}
s.Close();
```

Работа с разными кодировками.

По умолчанию в .NET все текстовые данные в кодировке UTF8, но часто требуется считать текстовый файл, сохраненный в другой кодировке, допустим в WIN1251. В таком случае если в файле был русский текст в кодировке WIN1251, при считывании его в UTF8 мы получим нечитаемые данные. Для того чтобы переводить строки из одной кодировки в другую существует класс Encoding из пространства имен System.Text, благодаря которому зная исходную кодировку мы можем привести текстовые данные к нужной кодировке. Рассмотрим следующий пример, например, мы хотим конвертировать считанную строку в кодировке WIN1251 в кодировку DOS(866), для этого мы можем использовать следующий код:

```
class Program
{
    private static string in1251;
    private static readonly Encoding enc1251 = Encoding.GetEncoding(1251);
    private static readonly Encoding enc866 = Encoding.GetEncoding(866);

    static void Main(string[] args)
    {
        //....
        //тут каким то образом получаем данные в in1251

        byte[] sourceBytes = enc1251.GetBytes(in1251);
        string outputString = enc866.GetString(sourceBytes);

        //далее делаем то что требуется с полученной строкой
        //....
    }
}
```

В sourceBytes мы получили входную строку в виде массива байт, которые далее мы можем так же посредством Encoding сохранить в строку с нужной кодировкой. В случае, когда мы читаем текст уже известной кодировке из файла, дела обстоят еще проще. При создании экземпляра StreamReader мы можем явно указать кодировку источника, код будет выглядеть следующим образом:

```
using (var sr = new StreamReader("Mytext.txt",
         Encoding.GetEncoding(1251)))
 {
     string read = null;
     while ((read = sr.ReadLine()) != null)
          Console.WriteLine(read);
 }
```

При записи в файл в нужной кодировке для **StreamWriter** так же можно указать кодировку, в которой будет текст.

Практическая часть

Общее задание: Написать класс который способен считывать из текстового файла массив (ступенчатый) и преобразовать его в новый массив, согласно заданию, записать его в файл. Первая строка в файле обязательно целое число — это количество строк в ступенчатом массиве

Вариант	Задание
1	1. Считать из файла input1.txt ступенчатый массив. Умножить минимальное (общее) на максимальное в каждой строке. Должен получится массив и из нового массива найти среднее арифметическое. Результат записать в файл output1.txt. записать полученный массив и число 2. Считать из файла input2.txt текст. Отсортировать предложения по проценту гласных букв в нем. Вывести вначале те, где процент меньше, а затем те где их больше.
2	1. Считать из файла input1.txt ступенчатый массив. Посчитать количество простых чисел в каждой строке. Результат записать в файл output1.txt. в виде (в первой строке 2 простых числа, во второй строке одно простое число и так далее (можно и вывести их, но это на доп). Каждый вывод на новой строке. 2. Считать из файла input2.txt текст. Вывести предложения в которых больше всего слов палиндромов(пример: шалаш – читаются одинаково в обе стороны)
3	1. Считать из файла input1.txt ступенчатый массив. Посчитать среднее в каждой строке, найти минимальное и максимальное также в каждой строке и получить новый массив (ср*(min+max)). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Отсортировать предложения по проценту знаков препинания и пробелов в нем. Вывести вначале те, где их меньше, а затем те где их больше.
4	1. Считать из файла input1.txt ступенчатый массив. Посчитать среднее арифметическое в каждой строке найти минимальное и среднее геометрическое получить массив, (ср.геом/ср.арифм+min). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести предложения в которых больше всего слов с заданной гласной.
5	1. Считать из файла input1.txt ступенчатый массив. Посчитать среднее арифметическое, чисел из которых можно взять квадратный корень в каждой строке найти минимальное и среднее геометрическое получить массив, (ср.геом/min – сум.ср.коней). Полученный массив записать в файл output1.txt.

	2. Считать из файла input2.txt текст. Вывести предложения в которых больше всего слов начинающихся и заканчивающихся с гласной.
6	1. Считать из файла input1.txt ступенчатый массив. Найти все минимальные в каждой строке. Получить массив (мин*факториал максимального числа ступенчатого массива / количество элементов в каждой строке). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести предложения в которых наименьшее количество знаков препинания.
7	1. Считать из файла input1.txt ступенчатый массив. Посчитать минимальное в каждой строке. Получить массив (мин+√сумвсехэлементов). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести предложения в которых больше всего слов с заданной согласной
8	1. Считать из файла input1.txt ступенчатый массив. Посчитать произведение в каждой строке и среднее арифметическое четных элементов, получить массив (произвед+ср.арифм). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести предложения в которых наименьшее количество знаков препинания.
9	1. Считать из файла input1.txt ступенчатый массив. Посчитать среднее арифметическое в каждой строке, получить массив (ср+max(всего ступ массива)-min(всего ступ массива)). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести строки в которых нет уникальных слов.
10	1. Считать из файла input1.txt ступенчатый массив. Посчитать среднее геометрическое в каждой строке, получить массив (ср+max(всего ступ массива)-min(всего ступ массива)). Полученный массив записать в файл output1.txt. 2. Считать из файла input2.txt текст. Вывести строки, которые состоят только из уникальных слов.

Контрольные вопросы:

1. Абстракции для работы с диском.
2. Абстракции для работы с каталогом.
3. Абстракции для работы с файлами.
4. Что такое поток? И какие абстракции есть для работы с ним.
5. Какая кодировка используется по умолчанию, как это изменить.
6. Перечисление FileMode.