**Лабораторная работа № 9 «Работа с файлами»**

**Цель и порядок работы**

Цель работы – изучить принципы работы с файлами средствами языка C sharp.

**Порядок выполнения работы:**

* ознакомиться с описанием лабораторной работы;
* выполнить задания;
* разработать схему алгоритма решения задачи;
* написать программу, соответствующую схеме алгоритма;
* ввести программу, отладить и выполнить ее на ЭВМ;

**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:**  
- cоздание файлов;  
- удаление файлов;  
- чтение данных;  
- запись данных;  
- изменение параметров файла (имя, расширение…);  
- другое.  
  
В Си-шарп 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;

**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 ()"); //допишет текст в конец файла  
}

**Как удалить файл?**  
  
Метод Delete() удаляет файл по указаному пути:

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

Кроме того, чтобы читать/записывать данные в файл с Си-шарп 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 Си-шарп. 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:\\file1.txt", FileMode.CreateNew); //создание нового файла  
   FileStream file2 = new FileStream("d:\\file2.txt", FileMode.Open); //открытие существующего файла  
   FileStream file3 = new FileStream("d:\\file3.txt", FileMode.Append); //открытие файла 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.  
  
**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.  
  
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.Create); //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  
}

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.Open);  
   StreamReader reader = new StreamReader(file1, Encoding.Unicode);  
   StreamWriter writer = new StreamWriter(file1, Encoding.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"); //указываем путь к файлу, а не поток  
   writer.WriteLine("текст");  
   writer.Close();  
}

**Как создать папку?**  
  
С помощью статического метода **CreateDirectory**() класса **Directory:**

static void Main(string[] args)  
{  
**Directory**.CreateDirectory("d:\\new\_folder");  
}

**Как удалить папку?**  
  
Для удаления папок используется метод Delete():

static void Main(string[] args)  
{  
   Directory.Delete("d:\\new\_folder"); //удаление 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  
}

**Задания**

**Задача 1.** Создайте файл numbers.txt и запишите в него натуральные числа от 1 до 500 через запятую.  
 **Задача 2.** Дан массив строк: "red", "green", "black", "white", "blue". Запишите в файл элементы массива построчно (каждый элемент в новой строке).

**Задача 3.** Возьмите любой текстовый файл, и найдите в нем размер самой длинной строки.

**Задача 4.** Ввести числа в файл. Найти максимум, минимум и посчитать количество отрицательных элементов в файле.

**Задача 5.** Составить программу записи в файл F1 массива 40 действительных чисел по убыванию, а в файл  F2 - по возрастанию. Массив задается в диапазоне от -10 до +10 случайным образом.

**Задача 6.** Файл  f1  содержит 120 чисел в 4 колонки. Создать программу считывания этих чисел и их  вывода в файл  f2  в 5 колонок.