

Лабораторная работа №15

Тема: Разработка программ создания и обработки файлов.

Цель: Научиться разрабатывать программы создания и обработки файлов.

Теоретические сведения:

Если вы хотите открыть текстовый файл, соответствующий поток данных будет заключен в оболочку из текстового декодера `StreamReader` или `StreamWriter`. Рассмотрим примеры записи и чтения текстовых файлов.

//Не забудьте прописать в начале файла следующий код:

```
using System.IO;
```

```
// Пример записи в файл "temp.txt"
```

```
string destFilename = "temp.txt";
```

```
using (StreamWriter writer = File.CreateText(destFilename))
```

```
{
```

```
writer.WriteLine("Записываемый текст");
```

```
}
```

//Не забудьте прописать в начале файла следующий код:

```
using System.IO;
```

```
// Пример чтения информации из файла и вывода ее на экран
```

```
string destFilename = "temp.txt";
```

```
using (StreamReader reader = File.OpenText(destFilename))
```

```
{
```

```
string line = null;
```

```
do
```

```
{ line = reader.ReadLine();
```

```
Console.WriteLine(line);
```

```
} while (line != null); }}
```

Для записи в текстовый файл используется класс `StreamWriter`. Некоторые из его конструкторов, которые могут применяться для создания объекта `StreamWriter`:

- `StreamWriter(string path)`: через параметр `path` передается путь к файлу, который будет связан с потоком

- `StreamWriter(string path, bool append)`: параметр `append` указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно `true`, то новые данные добавляются в конец файла. Если равно `false`, то файл перезаписывается заново

– StreamWriter(string path, bool append, System.Text.Encoding encoding): параметр encoding указывает на кодировку, которая будет применяться при записи

Свою функциональность StreamWriter реализует через следующие методы:

- int Close(): закрывает записываемый файл и освобождает все ресурсы
- void Flush(): записывает в файл оставшиеся в буфере данные и очищает буфер.
- Task FlushAsync(): асинхронная версия метода Flush
- void Write(string value): записывает в файл данные простейших типов, как int, double, char, string и т.д. Соответственно имеет ряд перегруженных версий для записи данных элементарных типов, например, Write(char value), Write(int value), Write(double value) и т.д.
- Task WriteAsync(string value): асинхронная версия метода Write
- void WriteLine(string value): также записывает данные, только после записи добавляет в файл символ окончания строки
- Task WriteLineAsync(string value): асинхронная версия метода WriteLine.

Пример 1. Запись в файл.

```
using System;
using System.IO;

namespace HelloApp
{
    class Program
    {
        static void Main(string[] args)
        {
            string writePath = @"C:\SomeDir\hta.txt";

            string text = "Привет мир!\nПока мир...";
            try
            {
                using (StreamWriter sw = new
StreamWriter(writePath, false, System.Text.Encoding.Default))
                {
                    sw.WriteLine(text);
                }

                using (StreamWriter sw = new
StreamWriter(writePath, true, System.Text.Encoding.Default))
```

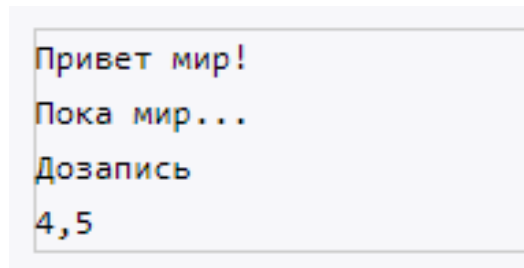
```

        {
            sw.WriteLine("Дозапись");
            sw.Write(4.5);
        }
        Console.WriteLine("Запись выполнена");
    }
    catch (Exception e)
    {
        Console.WriteLine(e.Message);
    }
}
}
}

```

В данном случае два раза создаем объект `StreamWriter`. В первом случае если файл существует, то он будет перезаписан. Если не существует, он будет создан. И в нее будет записан текст из переменной `text`. Во втором случае файл открывается для дозаписи, и будут записаны атомарные данные - строка и число. В обоих случаях будет использоваться кодировка по умолчанию.

По завершении программы в папке `C://SomeDir` мы сможем найти файл `hta.txt`, который будет иметь следующие строки:



```

Привет мир!
Пока мир...
Дозапись
4,5

```

Рисунок 1 – Результат выполнения программы.

Поскольку операции с файлами могут занимать продолжительное время, то в общем случае рекомендуется использовать асинхронную запись.

Пример 2. Асинхронные версии методов:

```

using System;
using System.IO;
using System.Threading.Tasks;

namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string writePath = @"C:\SomeDir\hta2.txt";

            string text = "Привет мир!\nПока мир...";

```

```

        try
        {
            using (StreamWriter sw = new
StreamWriter(writePath, false, System.Text.Encoding.Default))
            {
                await sw.WriteLineAsync(text);
            }

            using (StreamWriter sw = new
StreamWriter(writePath, true, System.Text.Encoding.Default))
            {
                await sw.WriteLineAsync("Дозапись");
                await sw.WriteLineAsync("4,5");
            }
            Console.WriteLine("Запись выполнена");
        }
        catch (Exception e)
        {
            Console.WriteLine(e.Message);
        }
    }
}

```

Обратите внимание, что асинхронные версии есть не для всех перегрузок метода Write.

Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

Некоторые из конструкторов класса StreamReader:

- StreamReader(string path): через параметр path передается путь к считываемому файлу
- StreamReader(string path, System.Text.Encoding encoding): параметр encoding задает кодировку для чтения файла

Среди методов StreamReader можно выделить следующие:

- void Close(): закрывает считываемый файл и освобождает все ресурсы
- int Peek(): возвращает следующий доступный символ, если символов больше нет, то возвращает -1
- int 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

- Task<int> ReadAsync(): асинхронная версия метода Read
- string ReadLine(): считывает одну строку в файле
- string ReadLineAsync(): асинхронная версия метода ReadLine
- string ReadToEnd(): считывает весь текст из файла
- string ReadToEndAsync(): асинхронная версия метода ReadToEnd

Пример 3. Сначала считаем текст полностью из ранее записанного файла:

```
using System;
using System.IO;
using System.Threading.Tasks;

namespace HelloApp
{
    class Program
    {
        static async Task Main(string[] args)
        {
            string path = @"C:\SomeDir\hta.txt";

            try
            {
                using (StreamReader sr = new StreamReader(path))
                {
                    Console.WriteLine(sr.ReadToEnd());
                }
                // асинхронное чтение
                using (StreamReader sr = new StreamReader(path))
                {
                    Console.WriteLine(await sr.ReadToEndAsync());
                }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Считаем текст из файла построчно:

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

```

using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
{
    string line;
    while ((line = sr.ReadLine()) != null)
    {
        Console.WriteLine(line);
    }
}
// асинхронное чтение
using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
{
    string line;
    while ((line = await sr.ReadLineAsync()) != null)
    {
        Console.WriteLine(line);
    }
}

```

В данном 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.

Индивидуальные задания:

Задание 1. Модернизируйте программу из Лабораторной работы №8 так, чтобы текст, предназначенный для обработки, читался из файла.

Задание 2. В соответствии со своим вариантом (номер варианта по списку подгруппы) разработать программу для работы с файлами на языке C#. Предварительно создать текстовый файл F1 не менее чем из 10 строк и записать в него информацию.

Таблица 1 – Варианты задания

№ варианта	Условие задачи
1	Скопировать в файл <i>F2</i> только четные строки из <i>F1</i> . Подсчитать размер файлов <i>F1</i> и <i>F2</i> (в байтах).
2	Скопировать в файл <i>F2</i> только те строки из <i>F1</i> , которые начинаются с буквы «А». Подсчитать количество слов в <i>F2</i> .
3	Скопировать из файла <i>F1</i> в файл <i>F2</i> строки, начиная с <i>K</i> до <i>K+5</i> . Подсчитать количество гласных букв в <i>F2</i> .

№ варианта	Условие задачи
4	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, которые не содержат цифры. Подсчитать количество строк, которые начинаются на букву «А» в файле <i>F2</i> .
5	Скопировать из файла <i>F1</i> в файл <i>F2</i> строки, начиная с четвертой по порядку. Подсчитать количество символов в последнем слове <i>F2</i> .
6	Скопировать из файла <i>F1</i> в файл <i>F2</i> строки, начиная с <i>N</i> до <i>K</i> . Подсчитать количество согласных букв в файле <i>F2</i> .
7	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, которые содержат только одно слово. Найти самое длинное слово в файле <i>F2</i> .
8	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, кроме той строки, в которой больше всего гласных букв. Напечатать номер этой строки.
9	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, начинающиеся на букву «А», расположенные между строками с номерами <i>N1</i> и <i>N2</i> . Определить количество слов в первой строке файла <i>F2</i> .
10	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, в которых нет слов, совпадающих с первым словом. Определить количество согласных букв в первой строке файла <i>F2</i> .
11	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, которые содержат только одно слово. Найти самое короткое слово в файле <i>F2</i> .
12	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, в которых есть слова, совпадающие с первым словом. Определить количество согласных букв в последней строке файла <i>F2</i> .
13	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, в которых более 2 слов. Определить номер слова, в котором больше всего гласных букв.
14	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, в которых содержится два одинаковых слова. Определить номер слова, в котором больше всего букв «А».
15	Скопировать из файла <i>F1</i> в файл <i>F2</i> все строки, в которых содержится не менее двух одинаковых слов. Определить номер слова, в котором больше всего цифр.

№ варианта	Условие задачи
16	Скопировать из файла F1 в файл F2 все строки, в которых есть слова, совпадающие со вторым словом. Определить количество цифр в последней строке файла F2.

Шкала оценивания индивидуальных заданий

Примеры	1-4 баллов
Примеры + Задание 1	1-7 баллов
Примеры + Задание 1+ Задание 2	1-10 баллов

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. При выполнении примеров, необходимо в отчет внести скриншоты готовых программ.
5. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
6. Вывод по лабораторной работе.