ХРАНЕНИЕ ИНФОРМАЦИИ В ФАЙЛАХ. НА ПРИМЕРЕ C#.

САМОЙЛОВ МИХАИЛ ЮРЬЕВИЧ

ФАЙЛЫ

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.

РАБОТА С ФАЙЛАМИ В C#

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`.

КЛАСС FILE

- File.Create("C:\\my_file.txt");
- File.WriteAllText("C:\\my_file.txt", "текст");
- File.AppendAllText("C:\\my_file.txt", "еще текст");
- string lines = File.ReadAllText("C:\\my_file.txt");
- File.Delete("C:\\my_file.txt");

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.**

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

ФАЙЛОВЫЙ ПОТОК

```
FileStream file = new FileStream("C:\\my_file.txt", FileMode.Open, FileAccess.Read);
```

РЕЖИМ ДОСТУПА

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.

РЕЖИМЫ ОТКРЫТИЯ

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

ЧТЕНИЕ ИЗ ФАЙЛА

Для чтения данных из потока нам понадобится класс **StreamReader**. В нем реализовано множество методов для удобного считывания данных.

ЧТЕНИЕ ИЗ ФАЙЛА

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*.

ЧТЕНИЕ ИЗ ФАЙЛА

```
FileStream file1 = new FileStream("C:\\my_file.txt", FileMode.Open);

StreamReader reader = new StreamReader(file1);

string lines = reader.ReadToEnd();

reader.Close();
```

ЗАПИСЬ В ФАЙЛ

Для записи данных в поток используется класс **StreamWriter**.

ЗАПИСЬ В ФАЙЛ

```
FileStream file1 = new FileStream("C:\\my_file.txt", FileMode.Create);
StreamWriter writer = new StreamWriter(file1);

writer.Write("текст");
writer.Close();
```

ЗАПИСЬ В ФАЙЛ

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

ПОТОКИ

При использовании StreamReader и StreamWriter можно не создавать отдельно файловый поток FileStream, а сделать это сразу при создании StreamReader/StreamWriter

ПОТОКИ

```
StreamWriter writer = new StreamWriter("C:\\my_file.txt");
writer.WriteLine("текст");
writer.Close();
```

ПАПКИ

- Directory.CreateDirectory("C:\\my_folder");
- Directory.Delete("C:\\my_folder");
- Directory.Delete("C:\\my_folder", true);

ЗАДАЧИ

Задача 1. Создайте файл numbers.txt и запишите в него натуральные числа от 1 до 50 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.