Тема 11. Файлы. Работа с файлами.

Учебные вопросы:

1. Введение.
2. Основы работы с файлами. Класс File.
3. Работа с каталогами. Класс Directory.

1. Введение.

Файл (англ. file) — **именованная** область данных на носителе информации, используемая как базовый объект взаимодействия с данными в операционных системах.

Работа с файлами играет ключевую роль в программировании, так как файлы являются одним из основных способов хранения и обмена данными. Многие приложения требуют сохранения данных на диск для последующего использования или передачи другим системам.

Основные операции с файлами.

Работа с файлами в программировании обычно включает несколько основных операций:

- **Создание файла**: Создание нового файла на диске для записи данных.
- **Чтение файла**: Получение данных из файла для использования в программе.
- **Запись в файл**: Запись данных в файл для их сохранения.
- **Удаление файла**: Удаление файла с диска, когда он больше не нужен.

Файл имеет следующие основные характеристики:

- **Имя файла**: Каждому файлу присваивается уникальное имя в рамках каталога (папки). Имя файла обычно включает расширение, указывающее тип файла (например, .txt, .jpg, .cs).
- **Тип файла**: Файлы могут хранить разные типы данных, такие как текст, изображения, аудио, видео, бинарные данные и др.
- **Размер файла**: Файлы могут иметь различный размер в зависимости от объема данных, которые они содержат.
- **Местоположение** (путь): Файлы хранятся в каталогах (папках), и доступ к ним осуществляется через указание полного (относительного) пути.
- **Доступ к файлу**: Доступ к файлам регулируется операционной системой, которая может ограничивать права на чтение, запись или выполнение файлов для разных пользователей или программ.

Библиотеки и пространства имен.

System.IO - основное пространство имен для работы с файлами, потоками, и каталогами.

Классы:

- **File**: Статический класс для работы с файлами (создание, удаление, копирование и перемещение).
- **Directory**: Статический класс для работы с каталогами (создание, удаление, получение списка файлов и каталогов).
- **FileInfo**: Класс для работы с файлами через экземпляр объекта, что позволяет получать детальную информацию о файле.
- **DirectoryInfo**: Класс для работы с каталогами через экземпляр объекта.
- **StreamReader** и **StreamWriter**: Классы для работы с текстовыми потоками.
- **FileStream**: Класс для низкоуровневой работы с файлами, позволяющий читать и записывать данные по байтам.

System.Text: - пространство имен, включающее классы для работы с текстом и его кодировкой.

Классы:

- **Encoding**: Класс, предоставляющий различные способы кодирования символов, например, UTF8, ASCII.
- **StringBuilder**: Класс для создания и манипуляции строками, особенно полезен при построении больших строк в памяти перед записью в файл.

2. Основы работы с файлами. Класс File.

File — это статический класс, который предоставляет методы для выполнения операций с файлами. Подходит для простых операций с файлами, когда не требуется хранить состояние файла между вызовами методов.

Достоинства:

- Простота использования: Класс File предоставляет простой и интуитивно понятный интерфейс для выполнения основных операций с файлами.
- Статические методы: Все методы класса File являются статическими, что позволяет использовать их без создания экземпляра класса. Это делает код более компактным и читабельным.
- Безопасность доступа: Класс File автоматически открывает и закрывает файлы, обеспечивая безопасный доступ к ним.
- Поддержка универсальных путей к файлам: Класс File поддерживает использование универсальных путей к файлам, которые работают на разных платформах (Windows, Linux, macOS).
- Широкий функционал: Класс File предоставляет множество полезных методов для работы с файлами, таких как Exists(), Delete(), Copy(), Move() и т.д.

Недостатки:

- Ограниченность возможностей: Класс File предназначен для выполнения базовых операций с файлами. Для более сложных задач, таких как управление потоками, работа с атрибутами файлов и каталогов, необходимо использовать другие классы, например, FileStream, FileInfo и Directory.
- Отсутствие гибкости: Поскольку методы класса File являются статическими, они не позволяют использовать дополнительные параметры, которые могут потребоваться для некоторых операций.
- Отсутствие транзакционности: Класс File не поддерживает транзакционность, что может быть важным в некоторых сценариях, где необходимо гарантировать целостность данных.
- Отсутствие поддержки асинхронности: Большинство методов класса File являются синхронными, что может привести к блокировке потока при выполнении длительных операций с файлами. Для асинхронных операций необходимо использовать другие классы, такие как FileStream.

Основные методы класса File

- **Create(string path).** Создаёт новый файл. Если файл уже существует, он будет перезаписан.

```
string path = "example.txt";
File.Create(path).Close(); // Создает файл и закрывает поток.
```

- **File.Delete(string path).** Удаляет указанный файл.

```
string path = "example.txt";
if (File.Exists(path))
{
    File.Delete(path);
    Console.WriteLine($"{path} был удален.");
}
```

- **File.Exists(string path).** Проверяет, существует ли файл по указанному пути.

```
string path = "example.txt";
if (File.Exists(path))
{
    Console.WriteLine($"{path} существует.");
}
else
{
    Console.WriteLine($"{path} не существует.");
}
```

- **File.Copy(string sourceFileName, string destFileName).** Копирует файл в новое место.

```
string sourcePath = "example.txt";
string destPath = "copy_of_example.txt";
if (File.Exists(sourcePath))
{
    File.Copy(sourcePath, destPath, overwrite: true);
    Console.WriteLine($"{sourcePath} был скопирован в {destPath}.");
}
```

- **File.Move(string sourceFileName, string destFileName).**

Перемещает файл в новое место.

- **ReadAllText(string path):** Читает весь текст из файла и возвращает его в виде строки.

```
string path = "example.txt";
if (File.Exists(path))
{
    string content = File.ReadAllText(path);
    Console.WriteLine("Содержимое файла:");
    Console.WriteLine(content);
}
```

- **WriteAllText(string path, string contents)**: Записывает строку в файл. Если файл уже существует, его содержимое будет перезаписано.

```
string path = "example.txt";
string content = "This is a new content.";
File.WriteAllText(path, content);
Console.WriteLine("Текст был записан в файл.");
```

- **AppendAllText(string path, string contents):** Добавляет текст в конец.

```
string path = "example.txt";
string additionalContent = "This is an additional line.";
File.AppendAllText(path, additionalContent + Environment.NewLine);
Console.WriteLine("Текст был добавлен в файл.");
```

- **File.ReadAllLines (string path).** Считывает все строки файла и возвращает массив строк.

```
string path = "example.txt";
if (File.Exists(path))
{
    string[] lines = File.ReadAllLines(path);
    Console.WriteLine("Содержимое файла построчно:");
    foreach (string line in lines)
    {
        Console.WriteLine(line);
    }
}
```

- **File.WriteAllLines(string path, array).** Записывает массив строк в файл. Каждая строка записывается в отдельную строку файла. Если файл уже существует, его содержимое будет перезаписано.

```
string path = "example.txt";
string[] lines = { "First line", "Second line", "Third line" };
File.WriteAllLines(path, lines);
Console.WriteLine("Массив строк был записан в файл.");
```

GetAttributes(string path): Возвращает атрибуты файла (например, является ли файл скрытым, системным).

```
string path = "example.txt";

FileAttributes attributes = File.GetAttributes(path);

Console.WriteLine($"Атрибуты файла: {attributes}");
```

3. Работа с каталогами. Класс Directory.

Каталог (или директория) обозначает структурированное хранилище файлов и других каталогов на диске или в файловой системе. Каталог можно рассматривать как папку, которая организует и группирует файлы по логическим разделам.

Класс **Directory** в C# предоставляет статические методы для работы с файловыми системами, в частности, для работы с каталогами (папками).

Этот класс позволяет создавать, удалять, перемещать каталоги, а также получать информацию о них, такой как список файлов и подкаталогов.

Основные возможности класса Directory:

Создание директорий:

- **Directory.CreateDirectory(string path)** - создает директорию по указанному пути.

Перемещение и удаление директорий:

- **Directory.Move(string sourceDirName, string destDirName)** - перемещает директорию.
- **Directory.Delete(string path, bool recursive)** - удаляет директорию. Если параметр recursive установлен в true, то удаляются все файлы и подкаталоги.

Получение текущей рабочей директории:

- **Directory.GetCurrentDirectory()** - возвращает текущую рабочую директорию.
- **Directory.SetCurrentDirectory(string path)** - устанавливает текущую рабочую директорию.

Получение информации о директориях:

- **Directory.Exists(string path)** - проверяет существование директории по указанному пути.
- **Directory.GetCreationTime(string path)** - получает время создания директории.
- **Directory.GetLastAccessTime(string path)** - получает время последнего доступа к директории.
- **Directory.GetLastWriteTime(string path)** - получает время последнего изменения директории.
- **Directory.GetDirectories(string path)** - возвращает массив путей к подкаталогам указанного каталога.
- **Directory.GetFiles(string path)** - возвращает массив путей к файлам в указанном каталоге.

- **Directory.CreateDirectory(string path)** - создает директорию по указанному пути.

```
string path = "NewFolder/SubFolder";
Directory.CreateDirectory(path);
Console.WriteLine($"Каталог {path} был создан.");
```

- **Directory.Delete(string path, bool recursive)** - удаляет директорию. Если параметр recursive установлен в **true**, то удаляются все файлы и подкаталоги.

```
Directory.Delete(path, recursive: true);
Console.WriteLine($"Каталог {path} был удален.");
```

Directory.Exists(string path) - проверяет существование директории по указанному пути.

```
string path = "NewFolder/SubFolder";
if (Directory.Exists(path))
{
    Console.WriteLine($"Каталог {path} существует.");
}
else
{
    Console.WriteLine($"Каталог {path} не существует.");
}
```

Directory.GetFiles(string path) - возвращает массив путей к файлам в указанном каталоге.

```
string path = "NewFolder/SubFolder";
if (Directory.Exists(path))
{
    string[] files = Directory.GetFiles(path);
    Console.WriteLine("Список файлов в каталоге:");
    foreach (string file in files)
    {
        Console.WriteLine(file);
    }
}
```

Directory.GetDirectories(string path) - возвращает массив путей к подкаталогам указанного каталога.

```
string path = "NewFolder";
if (Directory.Exists(path))
{
    string[] directories = Directory.GetDirectories(path);
    Console.WriteLine("Список подкаталогов:");
    foreach (string directory in directories)
    {
        Console.WriteLine(directory);
    }
}
```

Контрольные вопросы:

- Что такое файл и какие основные операции с ним вы знаете?
- Каковы основные характеристики файла?
- Что такое класс File и какие его основные методы? Назовите методы и опишите их назначение.
- Как проверить существование файла или директории по указанному пути?
- Какие методы класса File используются для создания, удаления и перемещения файлов? Приведите примеры использования.
- Как класс Directory помогает в работе с каталогами? Перечислите основные методы класса Directory.
- Как можно проверить существование файла или каталога в C#?
- Как создать и удалить директорию с помощью класса Directory?

Домашнее задание:

1. Повторить материал лекции.
2. Задача 1: Создание и запись в файл. Напишите программу, которая создает текстовый файл и записывает в него строку, введённую пользователем. Программа должна запросить у пользователя ввод строки. Запишите эту строку в новый файл с именем output.txt. Если файл уже существует, перезапишите его.

3. Задача 2: Чтение из файла. Напишите программу, которая читает содержимое текстового файла и выводит его на экран. Программа должна открыть файл input.txt. Прочитайте весь текст из файла и выведите его на экран. Если файл не существует, программа должна вывести сообщение об ошибке.

4. Задача 3: Создание директории. Напишите программу, которая создает новую директорию с указанным пользователем именем. Программа должна запросить у пользователя имя новой директории. Создайте директорию с указанным именем в текущем каталоге.

Материалы лекций:

https://github.com/ShViktor72/Education2025