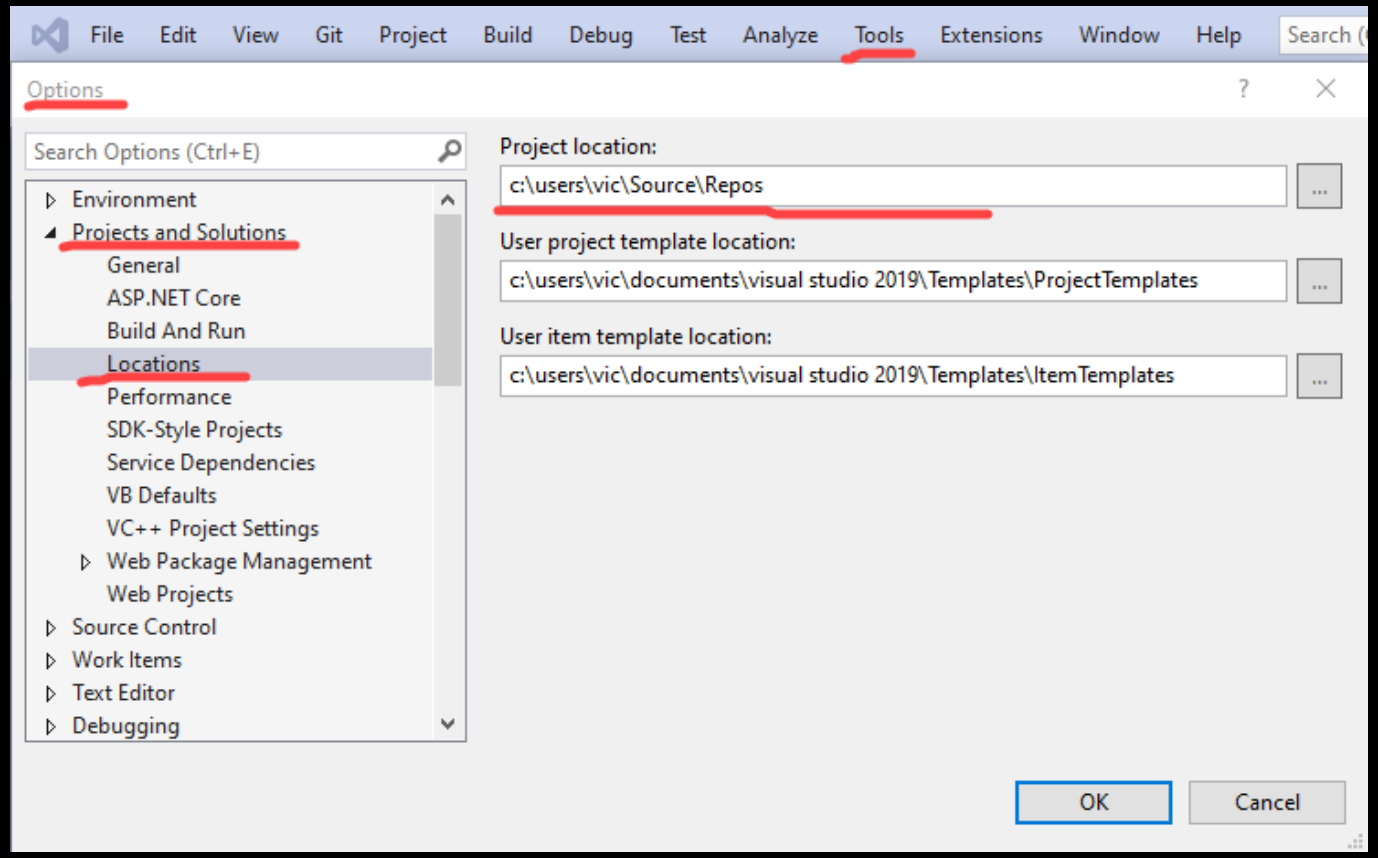
ЛЕКЦИЯ 12

- 11.10.2022
- Управление файлами и дисками
- Освобождение неуправляемых ресурсов

ЦЕЛИ ЛЕКЦИИ

- Вспомнить об организации файлов и папок во внешней памяти
- Познакомится со средствами языка C# по работе с дисковой системой
- Изучить некоторые виды исключения ввода-вывода
- Познакомится с using

КУДА VISUAL STUDIO СОХРАНЯЕТ ПРОЕКТ



ПОНЯТИЕ ФАЙЛА

Полное имя файла в MS–DOS и MS Windows:

```
Путь\имя_файла.расширение
```

Полное имя файла в UNIX:

```
Путь/имя_файла.расширение
```

Файл в каталоге исполняемым файлом проекта:

```
D:\Solution\ConsoleApplication1\bin\Debug\net5.0\file.txt
```

Примеры имен файлов:

C:\ProjectDir\myText.txt – текстовый файл;
D:\myExamples\example.exe – исполняемый файл;
C:\ProjectDir\myCode.cs – файл с исходным кодом на C#;
C:\myExamples\example.c – файл с исходным кодом на C.

ИМЕНА ФАЙЛОВ

В именах файлов в Windows запрещены символы:

> < | ? * / \ : "

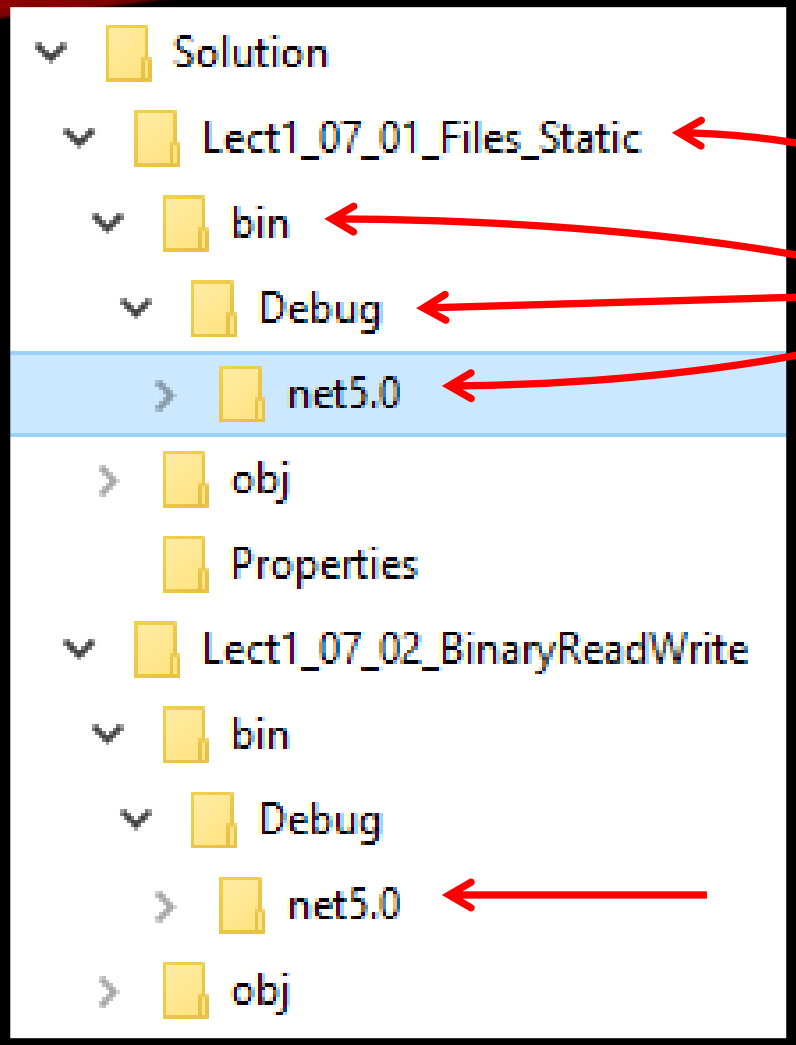
В именах файлов в Linux запрещён символ:

/

Дополнительно не рекомендуется использовать:

% # & { } $! ' @ + = Пробелы

ОТНОСИТЕЛЬНЫЕ ИМЕНА ФАЙЛОВ



Приложение запускается из
...Solution\ConsoleApplication1\bin\Debug:

"file.txt"

@"..\file.txt"

@"..\..\file.txt"

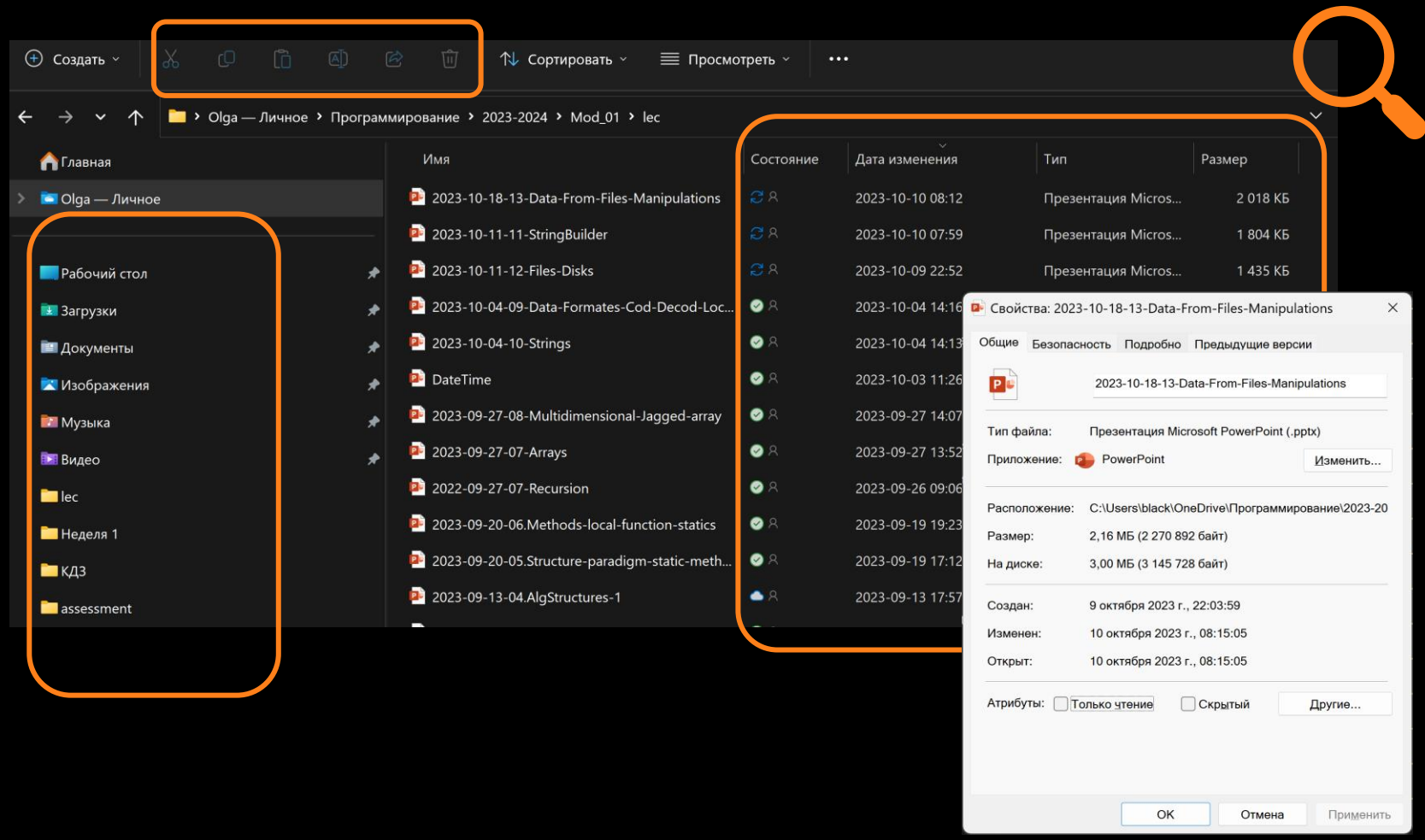
@"..\..\..\file.txt"

@"..\..\..\..\Lect1_07_02_BinaryReadWrite\bin\Debug\net5.0\file.txt"

@ – префикс буквального строкового литерала

ДЕЙСТВИЯ С КАТАЛОГАМИ И ФАЙЛАМИ

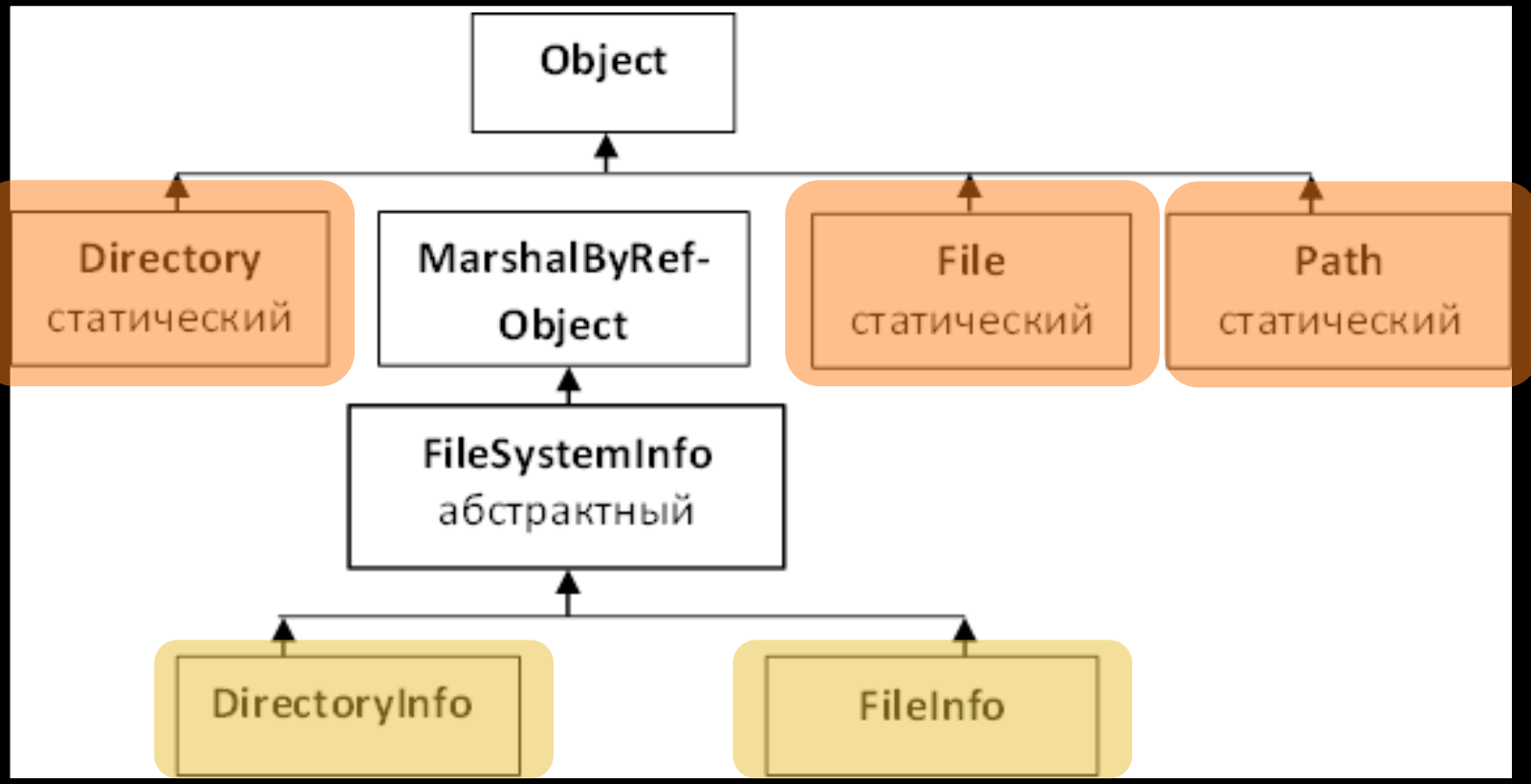
- Просмотр дерева каталогов
- Получение свойств каталогов и файлов
- Перемещение, копирование, создание, удаление каталогов и файлов
- Поиск каталогов и файлов
- Чтение данных из файлов и запись информации в файлы



СРЕДСТВА .NET ДЛЯ РАБОТЫ С ФАЙЛОВОЙ СИСТЕМОЙ

Пространство имён `System.IO`

```
using System.IO;
```



СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ КЛАССА FILE SYSTEM.IO

- Копирование, перемещение, переименование, создание, открытие, удаление и добавлению к файлу одной операцией записи
- Получение атрибутов файла или сведений, связанных с датой и временем создания, доступа, модификации
- Для групповых операций с файлами используют `Directory.GetFiles` или `DirectoryInfo.GetFiles`

ЧАСТО ВОЗНИКАЮЩИЕ ЗАДАЧИ ОБРАБОТКИ КАТАЛОГОВ

- Проверка существования каталога `Directory.Exists`
- Создание каталога `Directory.CreateDirectory`, `FileInfo.Directory`
- Создание вложенного каталога `DirectoryInfo.CreateSubdirectory`
- Переименование или перемещение каталога - `Directory.Move`, `DirectoryInfo.MoveTo`
- Копирование каталога `System.IO.Directory`, `System.IO.DirectoryInfo`
- Удаление каталога `Directory.Delete`, `DirectoryInfo.Delete`
- Получение имени и расширения файла `Path.GetFileName`
- Получение вложенных каталогов внутри данного `Directory.GetDirectories`, `DirectoryInfo.GetDirectories`
- Получение всех файлов и каталогов внутри данного `DirectoryInfo.EnumerateFileSystemInfos`
- Определение размера каталога `System.IO.Directory`

ЗАДАЧИ ОБРАБОТКИ ФАЙЛОВ

Действие	**Средства**
Создание текстового файла.	System.IO.File
Запись в текстовый файл.	System.IO.StreamWriter File.CreateText()
Чтение из текстового файла.	System.IO.StreamReader File.OpenText()
Добавление текста в файл.	File.AppendText( ) FileInfo.AppendText( )
Переименование или перемещение файла.	File.Move( ) FileInfo.MoveTo( )
Удаление файла	File.Delete( ) FileInfo.Delete( )
Копирование файла.	File.Copy( ) FileInfo.CopyTo( )
Получение размера файла.	FileInfo.Length
Получение атрибутов файла.	File.GetAttributes( )
Назначение атрибутов файла.	File.SetAttributes( )
Определение существования файла.	File.Exists( )
Чтение из двоичного файла.	System.IO.FileStream System.IO.BinaryReader
Запись в двоичный файл.	System.IO.FileStream System.IO.BinaryWriter
Извлечение расширения файла.	Path.GetExtension( )
Извлечение полного пути к файлу.	Path.GetFullPath( )
Извлечение имени и расширения файла из его пути.	Path.GetFileName( )
Изменение расширения файла.	Path.ChangeExtension( )

РАБОТА С ДИСКАМИ (DRIVEINFO)

Получение информации о физических дисках в системе:

```
public static System.IO.DriveInfo[] GetDrives();
```

Свойства DriveInfo	Краткое описание
AvailableFreeSpace	Указывает объем доступного свободного места на диске в байтах.
DriveFormat	Получает имя файловой системы, например NTFS или FAT32.
DriveType	Возвращает тип диска, например, компакт-диск, съемный, сетевой или несъёмный.
IsReady	Возвращает значение, указывающее, готов ли диск.
Name	Возвращает имя диска, например C:\.
RootDirectory	Возвращает корневой каталог диска.
TotalFreeSpace	Возвращает общий объем свободного места, доступного на диске, в байтах.
TotalSize	Возвращает общий размер места для хранения на диске в байтах.
VolumeLabel	Возвращает или задает метку тома диска.

ПРИМЕР

```
DriveInfo[] driveInfos = DriveInfo.GetDrives();
foreach(var driveInfo in driveInfos)
{
    if (driveInfo.IsReady)
    {
        Console.WriteLine($"Drive:: {driveInfo.Name} " +
            $"{Environment.NewLine}  type:: {driveInfo.DriveType}" +
            $"{Environment.NewLine}  size:: {driveInfo.TotalSize}" +
            $"{Environment.NewLine}  free:: {driveInfo.TotalFreeSpace}");
    }
}
```

Проверка готовности диска. При не готовности диска (false) обращение к нему вызовет исключение IOException

```
Drive:: C:\
  type:: Fixed
  size:: 976209571840
  free:: 843345854464
Drive:: D:\
  type:: Fixed
  size:: 1048951386112
  free:: 1047742926848
```

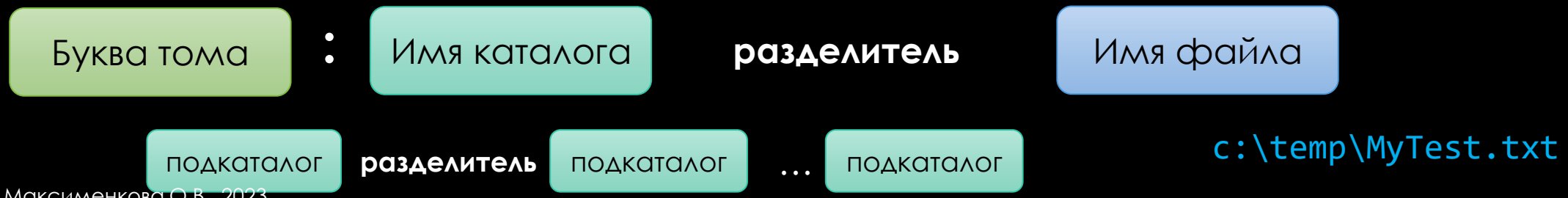
РАБОТА С ПУТЯМИ

Многие типы `System.IO` оперируют с путями при помощи операций класса `Path`

`Path` – статический класс, содержащий операции на объектами типа `String`, представляющие собой данные о путях к файлам и каталогам

Путь понимается, как строка, представляющая расположение файла или каталога

Формат пути определяется платформой, поэтому точное поведение членов класса Path зависят от платформы!



АБСОЛЮТНЫЙ И ОТНОСИТЕЛЬНЫЙ ПУТЬ

Абсолютные пути полностью указывают расположение: файл или каталог можно однозначно определить независимо от текущего расположения

Относительные пути указывают частичное расположение: текущее расположение используется в качестве отправной точки при поиске файла, указанного с относительным путем

```
string dir = Directory.GetCurrentDirectory();
Console.WriteLine($"Current directory is:: {dir}");
```

```
Current directory is:: D:\Programming\ConsoleApp1\ConsoleApp1\bin\Debug\net6.0
```

Большинство членов класса `Path` не проверяют наличие файлов, указанных в строке (исключение `Directory.Exists(path)`)

Абсолютный путь, разделители \

ПРИМЕРЫ ПУТЕЙ

Путь	Description
`C:\Documents\Newsletters\Summer2018.pdf`	Абсолютный путь к файлу из корня диска `C:`.
`\Program Files\Custom Utilities\StringFinder.exe`	Относительный путь от корня текущего диска.
`2018\January.xlsx`	Относительный путь к файлу в подкаталоге текущего каталога.
`..\Publications\TravelBrochure.pdf`	Относительный путь к файлу в каталоге, начиная с текущего каталога.
`C:\Projects\apilibrary\apilibrary.sln`	Абсолютный путь к файлу из корня диска `C:`.
`C:Projects\apilibrary\apilibrary.sln`	Относительный путь из текущего каталога диска `C:`.

Форматы путей к файлам в системах Windows - .NET | Microsoft Learn

КОНСТРУИРОВАНИЕ ПУТИ

```
string dir = Directory.GetCurrentDirectory();
Console.WriteLine($"Current directory is:: {dir}");

string[] pathItems = dir.Split('\\');
string restoredPath = "";

char separator = Path.DirectorySeparatorChar;
foreach(var pathItem in pathItems)
{
    restoredPath += pathItem + separator;
}
Console.WriteLine($"Current directory is:: {dir}");
```

Извлекаем актуальный символ-разделитель

Воссоздаём путь на основе полученного разделителя

ИСКЛЮЧЕНИЯ, СВЯЗАННЫЕ С ПУТЯМИ

NET Framework и версиях .NET Core 2.1+

.NET Core 2.1+

`ArgumentException`

все члены `Path`, которые принимают путь в качестве аргумента, если обнаруживают недопустимые символы пути

`GetFullPath` если строка содержит недопустимые символы пути

РАБОТА С КАТАЛОГАМИ: DIRECTORY И DIRECTORYINFO

`Directory` - статический класс с методами для создания, перемещения (копирования) и перечисления объектов (файлов и каталогов) в каталогах и вложенных каталогах

- `CreateDirectory()`
- `Delete()`
- `GetCurrentDirectory()`
- `SetLastAccessTime()`
- `SetCurrentDirectory()`
- `SetCreationTime()`

`DirectoryInfo` - класс с экземплярными методами для создания, перемещения (копирования) и перечисления объектов (файлов и каталогов) в каталогах и вложенных каталогах

Инструкция по задачам «Ввода и вывода» (Распространенные задачи ввода-вывода - .NET | Microsoft Learn)

ПРИМЕР С ЗАДАНИЕМ

```
string dir = Directory.GetCurrentDirectory();
Console.WriteLine("============Before adding=============");
Console.WriteLine($"Current directory is:: {dir}");
string[] dirs = null;
string[] files = null;
if (Directory.Exists(dir))
{
    dirs = Directory.GetDirectories(dir);
    files = Directory.GetFiles(dir);
}
Console.WriteLine("Directories::");
foreach (var d in dirs)
{
    Console.WriteLine(d);
}
Console.WriteLine("Files::");
foreach (var f in files)
{
    Console.WriteLine(f);
}
```

```
string dirName = dir + "tmp";
if (!Directory.Exists(dirName))
{
    Directory.CreateDirectory(dirName);
    Console.WriteLine($"Создан каталог: {dirName}");
}
else
{
    Console.WriteLine("Каталог уже существует!");
}
```

Код должен выводить каталоги и файлы папки, где лежит исполнимый файл, затем в этот каталог добавлять папку tmp и выводить новый список. Сейчас код работает некорректно. Исправьте это. Дополнительно печать массива строк оформите отдельным методом. Получение списка файлов и каталогов оформите отдельными методами

ПРИМЕР МЕТОДЫ КЛАССА DIRECTORY

```
using System;
using System.IO;

string dirName = Directory.GetCurrentDirectory();
Console.WriteLine($"Текущий каталог: \r\n{dirName}");
string[] dirNames = Directory.GetDirectories(@"..\..\..\");
Console.WriteLine("Каталоги выше на три уровня:");
foreach (string st in dirNames)
{
    Console.WriteLine($"Name: {st};\tCreationTime: " +
        Directory.GetCreationTime(st));
}
```

```
Текущий каталог:
D:\Примеры_программ\FileBegin\FileBegin_1\bin\Debug\net5.0
Каталоги выше на три уровня:
Name: ..\..\bin;   CreationTime: 10.10.2021 14:03:39
Name: ..\..\obj;   CreationTime: 10.10.2021 14:03:40
Name: ..\..\Properties;   CreationTime: 10.09.2021 14:03:40
```

РАБОТА С ФАЙЛАМИ: FILE И FILEINFO

`File` - статический класс с методами для создания, перемещения, копирования и открытия файлов

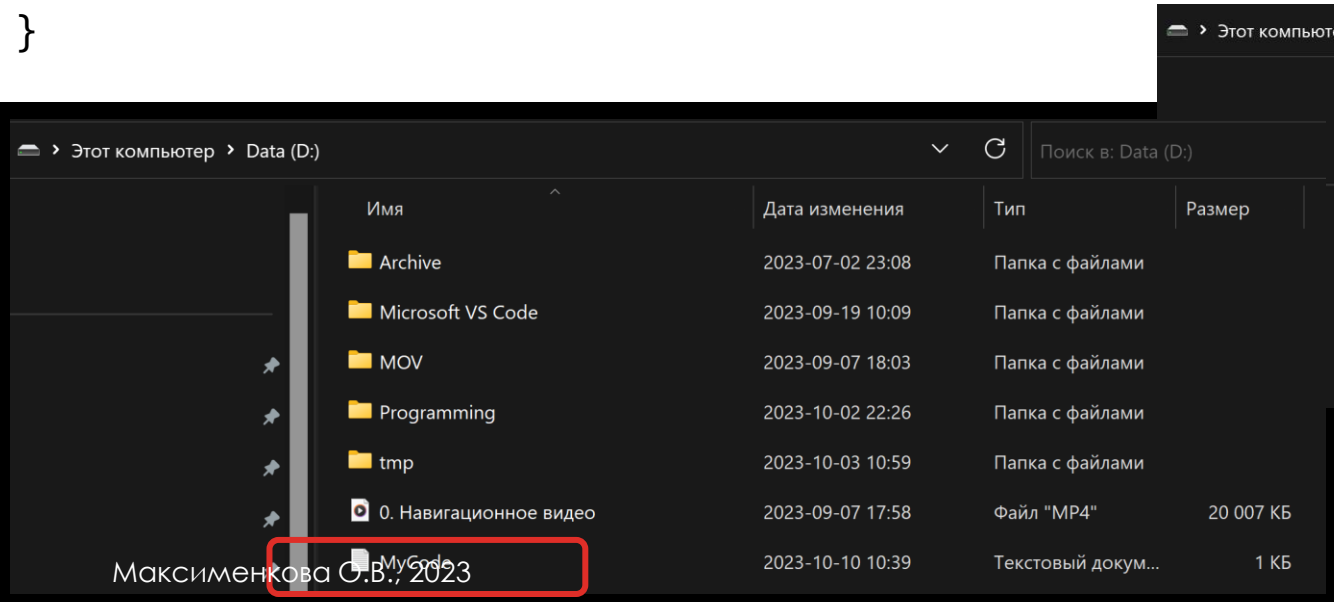
`DirectoryInfo` - класс с экземплярными методами для создания, перемещения , копирования и открытия файлов

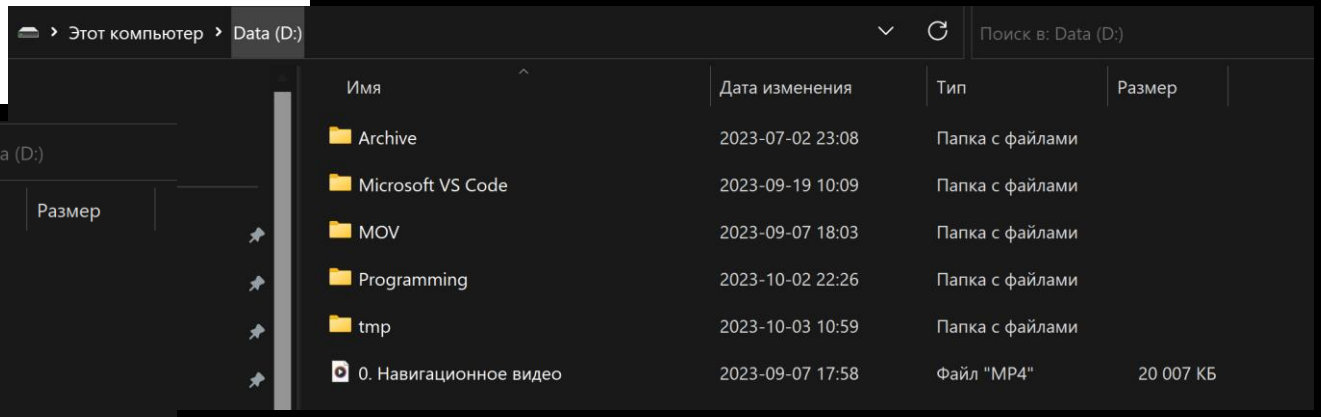
- По умолчанию всем пользователям предоставляется полный доступ на чтение и запись к новым файлам
- Управление поведением различных FileInfo методов реализована при помощи перечислений

Перечисление	Описание
FileAccess	Указывает доступ на чтение и запись к файлу.
FileShare	Указывает уровень доступа, разрешенный для файла, который уже используется.
FileMode	Указывает, сохраняется или перезаписывается содержимое существующего файла, а также вызывает ли запросы на создание существующего файла исключение.

ПРИМЕНЕНИЕ МЕТОДОВ КЛАССА FILE

```
if (!File.Exists("D:\\MyCode.txt"))
{
    File.Copy(@"../../../Program.cs", @"D:\MyCode.txt");
}
else
{
    Console.WriteLine("Файл D:\\MyCode.txt уже существует!");
}
```





ОСНОВНЫЕ МЕТОДЫ КЛАССОВ DIRECTORYINFO И FILEINFO

Имя	Описание
Create	Создает пустой файл или каталог с заданным параметром именем
Delete	Удаляет файл или каталог
MoveTo	Перемещает файл или каталог
CopyTo	Копирует существующий файл в новый (определён только в классе FileInfo)

ОТКРЫТЬ ФАЙЛ ДЛЯ СПЕЦИАЛЬНОЙ ОБРАБОТКИ

- FileStream OpenRead(string путь) – открыть существующий файл для чтения
- StreamReader OpenText(string путь) – открыть для чтения файл с текстом в кодировке UTF-8
- FileStream OpenWrite(string путь) – открыть существующий или создать новый файл для записи

Конструкторы DirectoryInfo и FileInfo:

```
DirectoryInfo(string путь)
FileInfo(string путь)
```

Пример:

```
FileInfo fileRef = new FileInfo(@"C:\file.txt");
```

ПРИМЕНЕНИЕ СРЕДСТВ КЛАССА FILEINFO

```
using System.IO;

const string refName = @"C:\const.txt";
FileInfo fileRef = new FileInfo(refName);

if (!fileRef.Exists)
{
    fileRef.Create();
}
```

СРЕДСТВА КЛАССОВ DIRECTORYINFO И FILEINFO: ПРИМЕР

```
using System;
using System.IO;

DirectoryInfo dirInf = new DirectoryInfo(@".");
Console.WriteLine($"*** Имя каталога: {dirInf.Name}");
Console.WriteLine("*** Имена и размеры файлов: ");
FileInfo[] files = dirInf.GetFiles();
foreach (FileInfo temp in files)
{
    Console.WriteLine($"{temp.Name}; Length = {temp.Length}");
}
```

Текущая папка

РАБОТА С СОДЕРЖИМЫМ ФАЙЛА

Чтение и запись

ЗАПИСЬ СРЕДСТВАМИ КЛАССА FILE

VOID WRITEALLTEXT(STRING PATH, STRING DATA [, SYSTEM.TEXT.ENCODING ENCODING]);

СОЗДАЕТ НОВЫЙ ФАЙЛ, ЗАПИСЫВАЕТ В НЕГО УКАЗАННУЮ СТРОКУ И ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ. ЕСЛИ ЦЕЛЕВОЙ ФАЙЛ УЖЕ СУЩЕСТВУЕТ, ОН БУДЕТ ПЕРЕОПРЕДЕЛЕН

VOID WRITEALLLINES(STRING PATH, STRING[] DATA [, SYSTEM.TEXT.ENCODING ENCODING]);

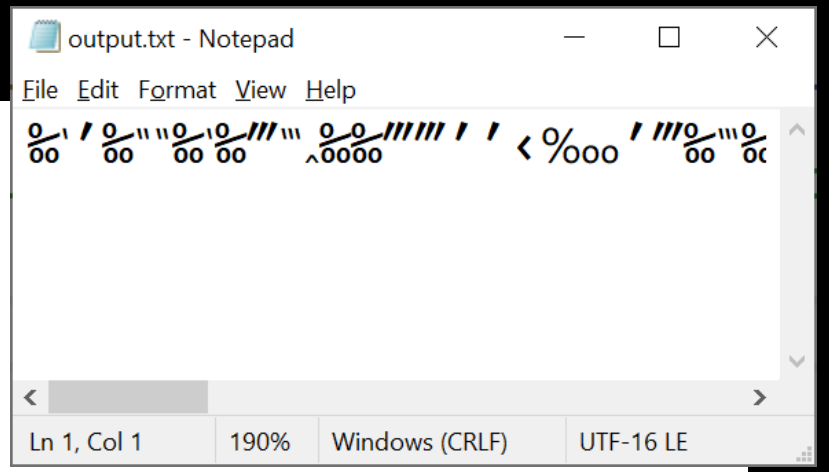
СОЗДАЕТ НОВЫЙ ФАЙЛ, ЗАПИСЫВАЕТ В НЕГО ОДНУ ИЛИ НЕСКОЛЬКО СТРОК, ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ

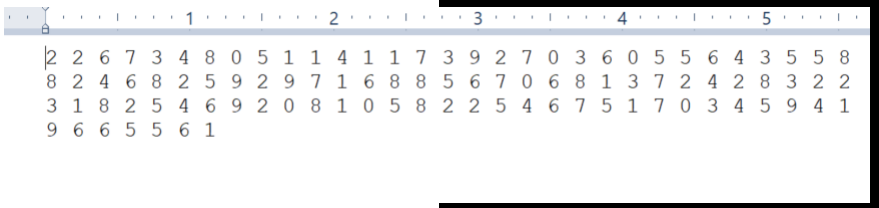
VOID WRITEALLBYTES(STRING PATH, BYTE[] DATA);

СОЗДАЕТ НОВЫЙ ФАЙЛ, ЗАПИСЫВАЕТ В НЕГО УКАЗАННЫЙ МАССИВ БАЙТОВ И ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ. ЕСЛИ ЦЕЛЕВОЙ ФАЙЛ УЖЕ СУЩЕСТВУЕТ, ОН БУДЕТ ПЕРЕОПРЕДЕЛЕН

ПРИМЕР: WRITEALLTEXT (1)

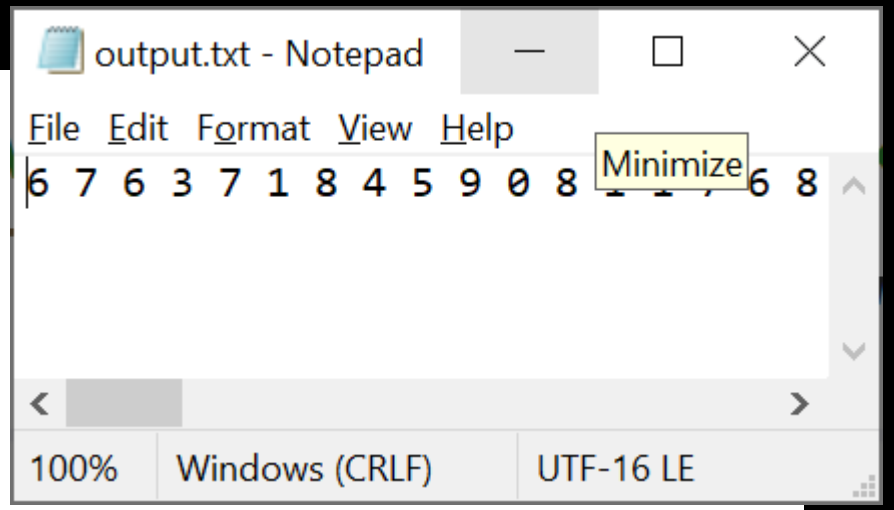
```
public class Program
{
    static Random rnd = new Random();
    public static void Main()
    {
        string dataToFile = string.Empty;

        for(int i = 0; i < 100; i++)
        {
           dataToFile += rnd.Next(10).ToString() + " ";
        }
        File.WriteAllText(@"output.txt", dataToFile);
    }
}
```





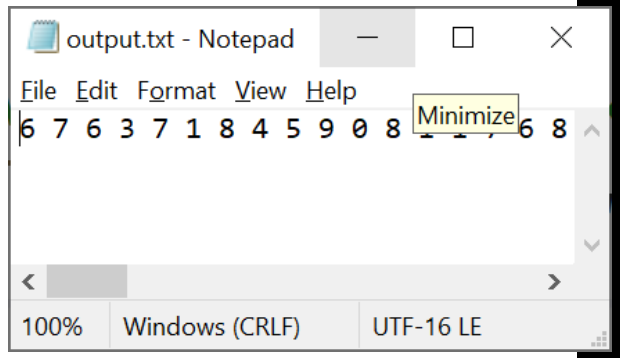
ПРИМЕР: WRITEALLTEXT (2)

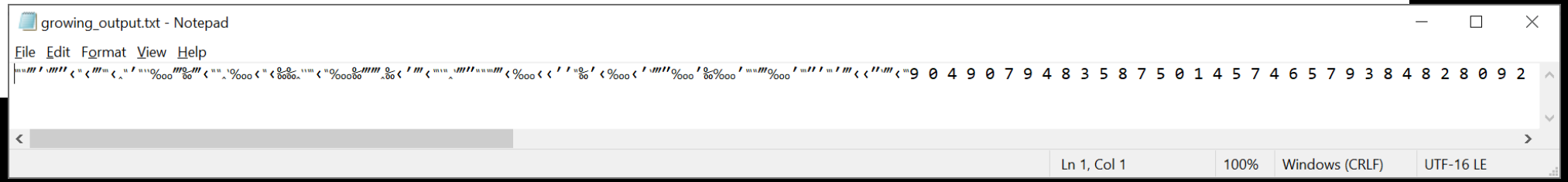
```
public class Program
{
    static Random rnd = new Random();
    public static void Main()
    {
        string dataToFile = string.Empty;

        for(int i = 0; i < 100; i++)
        {
           dataToFile += rnd.Next(10).ToString() + " ";
        }
        File.WriteAllText(@"output.txt", dataToFile, System.Text.Encoding.Unicode);
    }
}
```



ПРИМЕР: WRITEALLTEXT, APPENDALLTEXT

```
public class Program
{
    static Random rnd = new Random();
    public static void Main()
    {
        string dataToFile = string.Empty;

        for(int i = 0; i < 100; i++)
        {
           dataToFile += rnd.Next(10).ToString() + " ";
        }
        // File.WriteAllText(@"output.txt", dataToFile);
        File.WriteAllText(@"output.txt", dataToFile, System.Text.Encoding.Unicode);
        File.AppendAllText(@"growing_output.txt", dataToFile, System.Text.Encoding.Unicode);
    }
}
```





ЧТЕНИЕ СРЕДСТВАМИ КЛАССА FILE

STRING READALLTEXT(STRING PATH [, SYSTEM.TEXT.ENCODING ENCODING]);

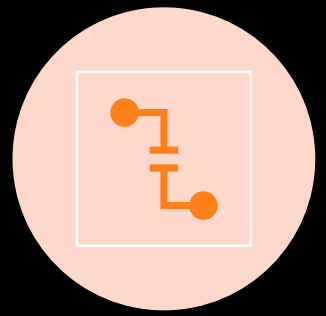
ОТКРЫВАЕТ ТЕКСТОВЫЙ ФАЙЛ, СЧИТЫВАЕТ ВЕСЬ ТЕКСТ ФАЙЛА В СТРОКУ И ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ

STRING[] READALLLINES(STRING PATH [, SYSTEM.TEXT.ENCODING ENCODING]);

ОТКРЫВАЕТ ТЕКСТОВЫЙ ФАЙЛ, СЧИТЫВАЕТ ВСЕ СТРОКИ ФАЙЛА В МАССИВ СТРОК И ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ

BYTE[] READALLBYTES(STRING PATH);

ОТКРЫВАЕТ ДВОИЧНЫЙ ФАЙЛ, СЧИТЫВАЕТ СОДЕРЖИМОЕ ФАЙЛА В МАССИВ БАЙТОВ И ЗАТЕМ ЗАКРЫВАЕТ ФАЙЛ



IENUMERABLE<STRING> READLINES(STRING PATH);

СЧИТЫВАЕТ СТРОКИ ФАЙЛА

Если файла не существует, генерируется исключение:
System.IO.FileNotFoundException

ПРИМЕР. ПОЛУЧЕНИЕ ВСЕХ СТРОК ФАЙЛА

```
string path = @"input.txt";
string dataFromFile = File.ReadAllText(path);
```

- Код завершится с исключением System.IO.FileNotFoundException, если файл не существует по указанному пути
- Код отработает корректно, если файл существует на диске

Чтобы код работал всегда, следует проверять наличие файла на диске, и сообщать пользователю о его отсутствии

```
try
{
    string dataFromFile = File.ReadAllText(path);
}
catch(System.IO.FileNotFoundException)
{
    Console.WriteLine($"File {path} doesn't exist, please check the path");
}
```

ПРИМЕР. НЕКОРРЕКТНЫЕ СИМВОЛЫ В ПУТИ

```
string path = @"??input.txt";
try
{
    string dataFromFile = File.ReadAllText(path);
}
catch(System.IO.FileNotFoundException)
{
    Console.WriteLine($"File {path} doesn't exist, please check the path");
}
```

- Если имя пути содержит недопустимые символы, появится исключение System.IO.IOException: The filename, directory name, or volume label syntax is incorrect.
- Метод `Path.GetInvalidPathChars()` возвращает массив, содержащий символы, которые не разрешены в именах путей

ПРИМЕР. ПРОВЕРКА КОРРЕКТНОСТИ ПУТИ

```
string path = @"??input.txt";
char[] invalidPathChars = Path.GetInvalidPathChars();
try
{
    if (invalidPathChars != null)
    {
        Console.WriteLine($"File path {path} contains invalid chars");
    }
    else
    {
        string dataFromFile = File.ReadAllText(path);
    }
}
catch(System.IO.FileNotFoundException)
{
    Console.WriteLine($"File {path} doesn't exist, please check the path");
}
```

Получаем массив недопустимых в пути символов

Помним, что здесь не проверяется автоматически, что путь ведёт к файлу и возможно исключение

Перехватываем его, в случае необходимости

READALLLINES() VS. READLINES

- `ReadAllLines()` – возвращает массив строк
  - [http://learn.microsoft.com/ru-ru/dotnet/api/system.io.file.readalllines?view=net-6.0]
- `ReadLines()` – возвращает итерируемую коллекцию, может использоваться в запросе
  - [http://learn.microsoft.com/ru-ru/dotnet/api/system.io.file.readlines?view=net-6.0]

ПОЛУЧЕНИЕ ДАННЫХ ИЗ ТЕКСТОВОГО ФАЙЛА

```
public static int[] GetDataFromFile(string path)
{
    int[] data = null;
    if (path.IndexOfAny(invalidPathChars) != -1 || File.Exists(path))
    {
        throw new FileNotFoundException();
    }
    string dataFromFile = File.ReadAllText(path);
    string[] dataFromString = dataFromFile.Split(" ", StringSplitOptions.RemoveEmptyEntries);
    data = new int[dataFromString.Length];
    for (int i = 0; i < dataFromString.Length; i++)
    {
        data[i] = int.Parse(dataFromString[i]);
    }
    return data;
}
```

using не требуется при такой реализации чтения из файла

USING ДЛЯ ОЧИСТКИ РЕСУРСОВ

```
static string ReadFile2(string fileName)
{
    using (System.IO.StreamReader reader =  System.IO.File.OpenText(fileName))
    {
        if (reader.EndOfStream)
            return null;
        return reader.ReadToEnd();
    }
}
```

`using` требуется для корректного использования объектов, реализующих `IDisposable`

`IDisposable` обязывает объект реализовать метод `Dispose` для явного освобождения неуправляемых ресурсов

ИСПОЛЬЗОВАННАЯ ЛИТЕРАТУРА

- https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks
- https://docs.microsoft.com/en-us/dotnet/api/system.io.fileshare?view=net-6.0
- DriveInfo Class (System.IO) | Microsoft Learn
- Path Класс (System.IO) | Microsoft Learn