В.В. Подбельский

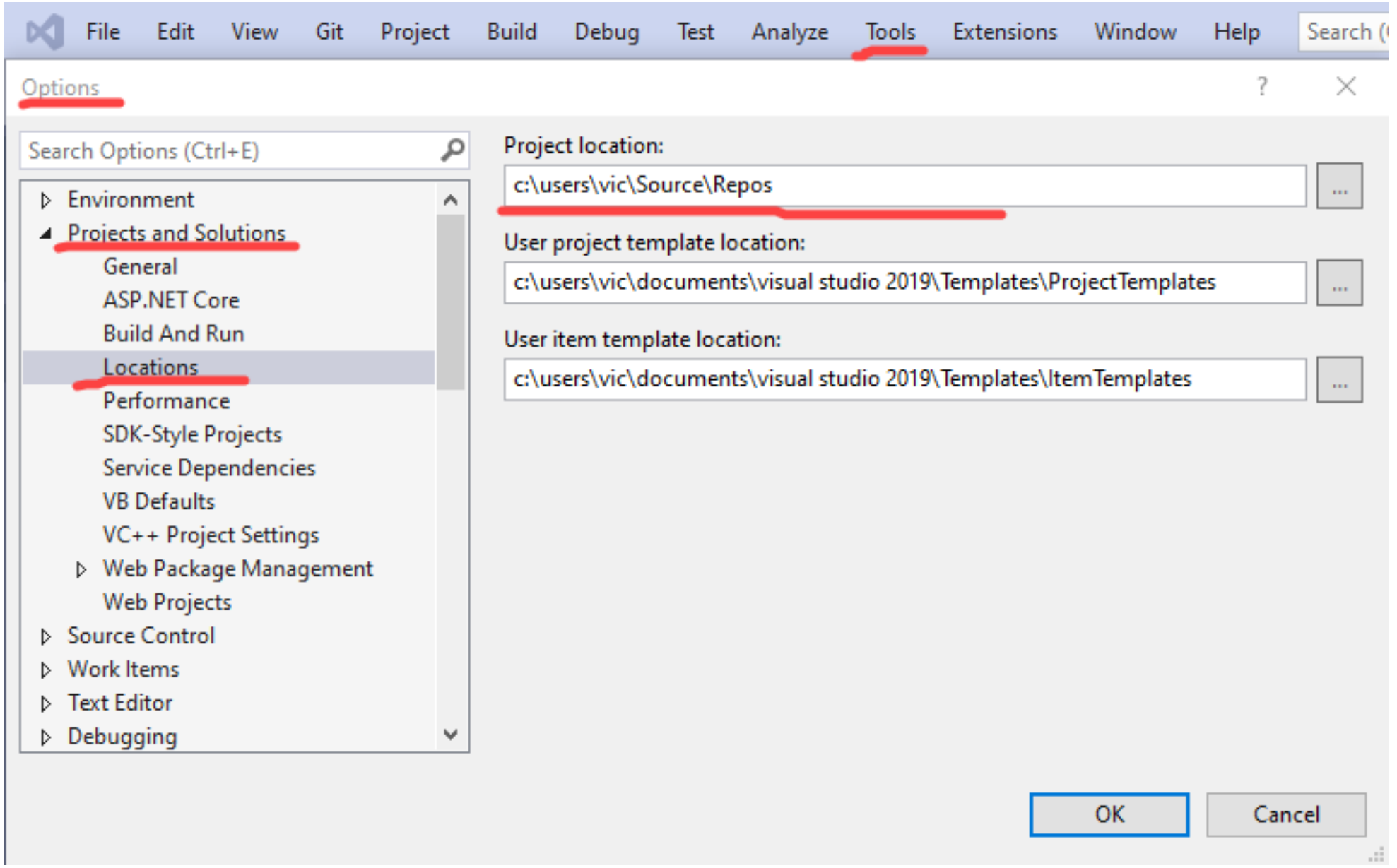
Иллюстрации к курсу лекций по дисциплине «Программирование на C#»

07

Статические Методы для Работы с Файловой Системой. Кодировки

Благодарность Д.Д. Сагалову за помощь в переработке презентаций

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 текстовый в каталоге с exe-модулем проекта:

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

Примеры имен файлов:

C:\ProjectDir\myText.txt – текстовый файл;

D:\myExamples\example.exe – исполнимый модуль;

C:\ProjectDir\myCode.cs – файл с исходным кодом на C#;

C:\myExamples\example.c – файл с исходным кодом на C.

Имена Файлов

В современных операционных системах присутствует набор символов, которые нельзя/не рекомендуется использовать в качестве имён файлов. Связано это с тем, что большинство этих символов выполняют служебные функции.

В именах файлов в Windows запрещены символы:
> < | ? * / \ : "

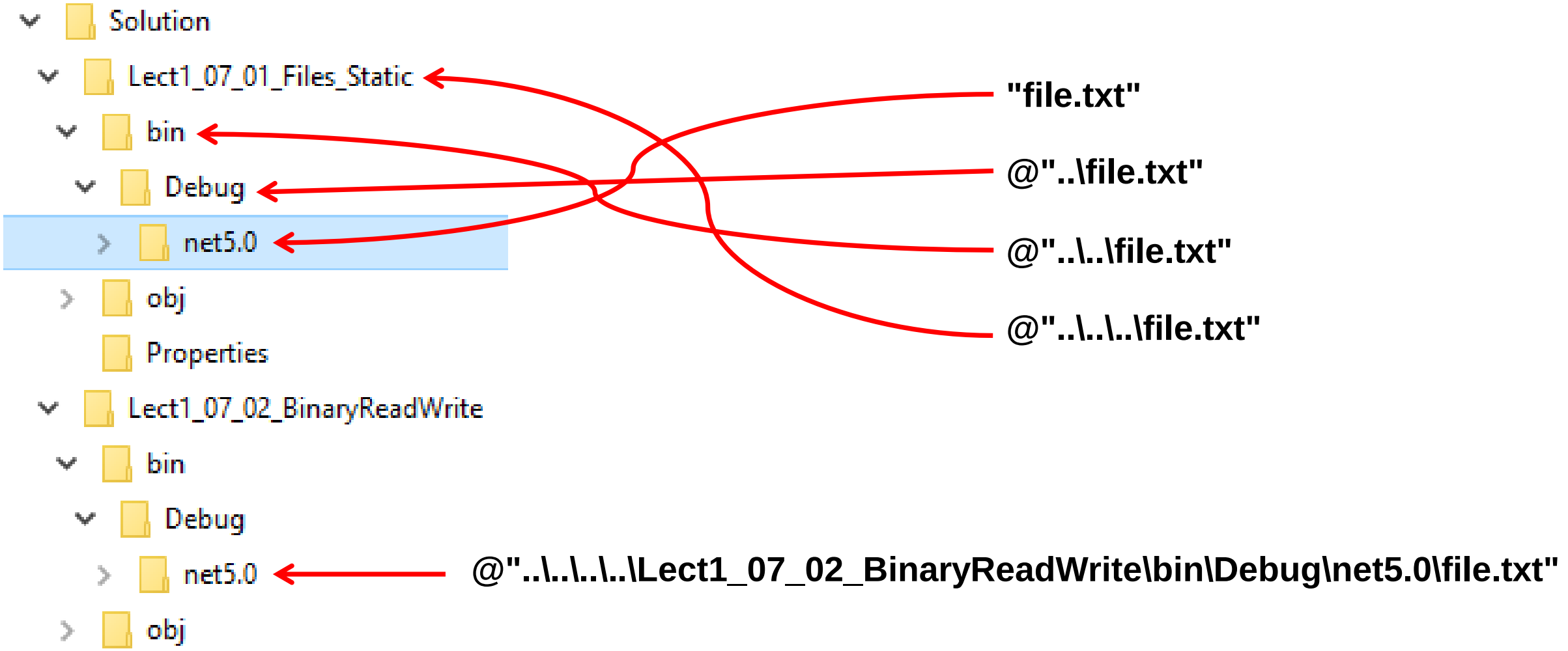
В именах файлов в Linux запрещён символ:
/

Дополнительно не рекомендуется использовать:
% # & { } $! ' @ + = Пробелы

https://en.wikipedia.org/wiki/Filename

Относительные Имена Файлов

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



Помните: @ – префикс буквального строкового литерала.

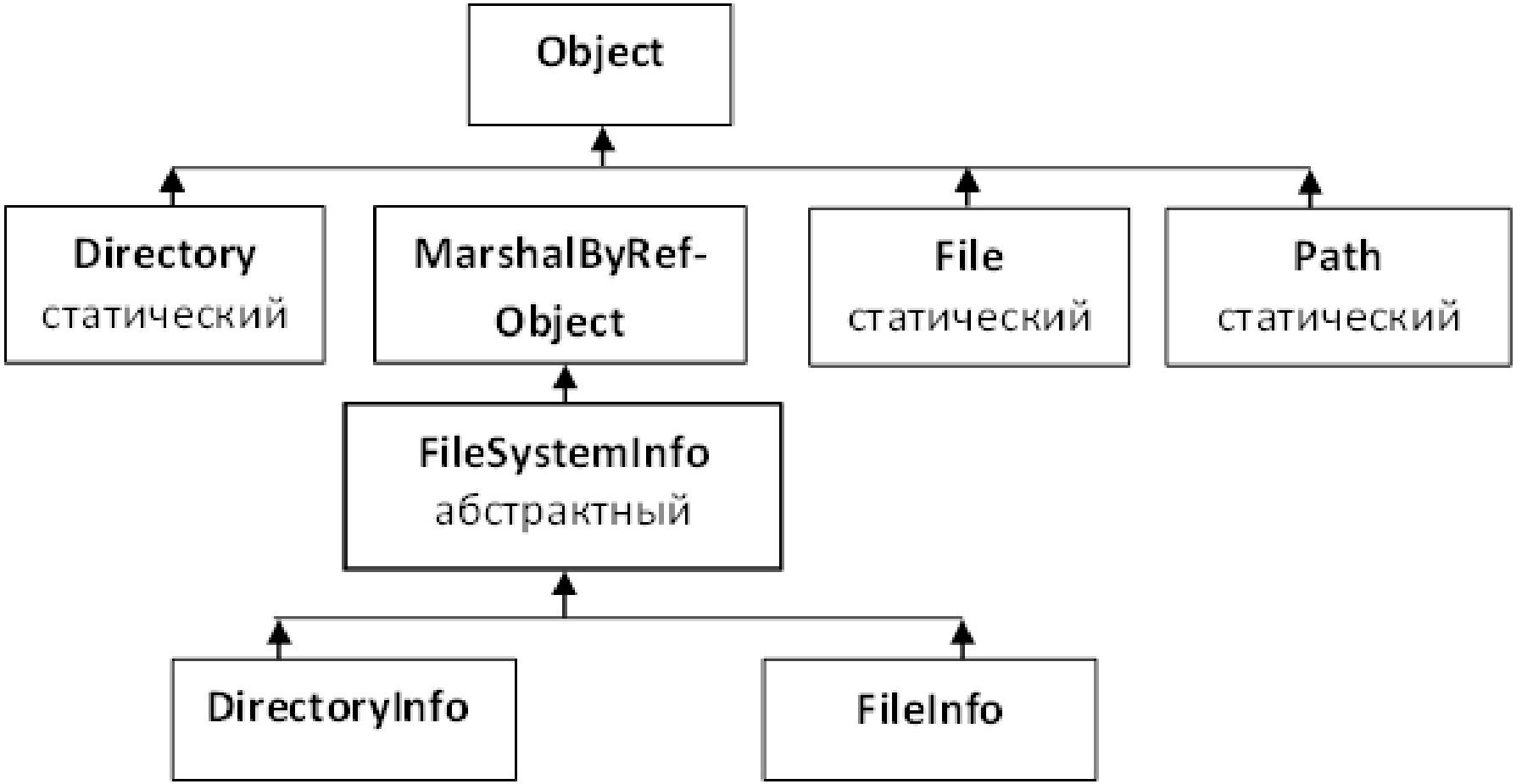
Действия с Каталогами и Файлами

Ниже перечислены наиболее типичные действия, которые приходится совершать при работе с файлами и каталогами:

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

Средства .NET для Работы с Файловой Системой

```
using System.IO;
```



Часто Возникающие Задачи Обработки Каталогов

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

https://docs.microsoft.com/en-us/dotnet/standard/io/common-i-o-tasks

Задачи обработки файлов

https://docs.microsoft.com/ru-ru/dotnet/standard/io/common-i-o-tasks

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)

Получить информацию о дисках в системе:

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

Свойства DriveInfo :

Свойства	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.

https://docs.microsoft.com/en-us/dotnet/api/system.io.driveinfo?view=net-5.0

Работа с Путями (Path)

public **static** class Path {…}

Поля (public static readonly):

char AltDirectorySeparatorChar	/
char DirectorySeparatorChar	\
char PathSeparator	;
char VolumeSeparatorChar	:

Методы:

ChangeExtension, **Combine**, EndsInDirectorySeparator,
GetDirectoryName, GetExtension, GetFileName, GetFileNameWithoutExtension,
GetFullPath, GetInvalidFileNameChars, GetInvalidPathChars, GetPathRoot,
GetRandomFileName, GetRelativePath, GetTempFileName,
GetTempPath, HasExtension, IsPathFullyQualified, IsPathRooted, Join,
TrimEndingDirectorySeparator, TryJoin

https://docs.microsoft.com/en-us/dotnet/api/system.io.path?view=net-5.0

Методы Класса Directory-1

```
DirectoryInfo CreateDirectory (string path)
void Delete (string path)
bool Exists (string path)

IEnumerable<string> EnumerateFiles(string path)
IEnumerable<string> EnumerateDirectories(string path)
IEnumerable<string> EnumerateFilesSystemEntries(string path)

string[] GetDirectories(string path)
string[] GetFiles(string path)
string[] GetFileSystemEntries(string path)
```

Методы Класса Directory-2

```
string GetCurrentDirectory()

DateTime GetCreationTime(string path)

void SetCreationTime(string path, DateTime time)

DateTime GetLastAccessTime(string path)

void SetLastAccessTime(string path, DateTime time)
```

Применение Методов Класса Directory: Пример 1

```
using System;
using System.IO;


string dirName = @"D:\Tempo";
if (!Directory.Exists(dirName))
{
    Directory.CreateDirectory(dirName);
    Console.WriteLine($"Создан каталог: {dirName}");
}
else
{
    Console.WriteLine("Каталог уже существует!");
}
```

Благодаря буквальным строковым литералам не приходится экранировать символы «\» в пути к файлу.

Применение Методов Класса Directory: Пример 2

```
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));
}
```

Результаты Выполнения Программы

Вывод:

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

Методы Класса File

```
void Copy(string source, string destination)
FileStream Create(string path)
StreamWriter CreateText(string path)
bool Exists(string path)

DateTime GetCreationTime(string path)
void SetCreationTime(string path, DateTime time)
DateTime GetLastAccessTime(string path)
void SetLastAccessTime(string path, DateTime time)
DateTime GetLastWriteTime(string path)
void SetLastWriteTime(string path, DateTime time)

FileStream Open(string path, FileMode mode
      [, FileAccess access
      [, FileShare sharing_mode ] ] )
```

Применение Методов Класса File

```
using System;
using System.IO;

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

Перечисление System.IO.FileAccess

Определяет константы для доступа к файлу для чтения/записи.

[FlagsAttribute] разрешает побитовое сочетание значений.

Имя	Значение	Краткое описание
Read	1	Доступ к файлу на чтение. Для получения доступа и для чтения, и для записи необходимо объединить с Write.
Write	2	Доступ к файлу на запись. Для получения доступа и для чтения, и для записи необходимо объединить с Read.
ReadWrite	3	Доступ и для чтения, и для записи. Данные можно записать в файл и/или прочитать из файла.

https://docs.microsoft.com/en-us/dotnet/api/system.io.fileaccess?view=net-5.0

Перечисление System.IO.FileShare

Определяет тип доступа, который другие процессы могут получить к тому же файлу. Помечено атрибутом [FlagsAttribute].

Имя	Значение	Описание
None	0	Запрещает совместное использование файла. Любой запрос на открытие файла (данным процессом или другим процессом) не выполняется до тех пор, пока файл не будет закрыт.
Read	1	Разрешает открытие файла для чтения. Помните, что даже если этот флаг задан, для доступа к данному файлу могут потребоваться дополнительные разрешения.
Write	2	Доступ и для чтения, и для записи. Данные можно записать в файл и/или прочитать из файла.
ReadWrite	3	Разрешает открытие файла для чтения и/или записи. Помните, что даже если этот флаг задан, для доступа к данному файлу могут потребоваться дополнительные разрешения.
Delete	4	Разрешает последующее удаление файла.
Inheritable	16	Разрешает наследование дескриптора файла дочерними процессами. В Win32 непосредственная поддержка этого свойства не обеспечена.

https://docs.microsoft.com/en-us/dotnet/api/system.io.fileshare?view=net-5.0

Перечисление System.IO.FileMode

Описывает, каким образом необходимо открывать файл.

Имя	Краткое описание
Append	Открыть существующий файл для дополнений или создать новый. Указатель позиции установить в конец потока.
Create	Создать новый файл для записи. Если существует одноименный, уничтожить.
CreateNew	Создать новый файл для записи. Если существует одноименный, генерируется исключение.
Open	Открыть существующий файл. Если файл отсутствует, генерируется исключение.
OpenOrCreate	Если файл существует, открыть его сохранив информацию. Иначе – создать новый.
Truncate	Открыть существующий файл, очистив его. Если файл отсутствует, генерируется исключение.

https://docs.microsoft.com/en-us/dotnet/api/system.io.filemode?view=net-5.0

Перечисление System.IO.FileAttributes

Описывает, какими особенностями обладает файл.

Имя	Описание
Compressed	Файл сжат.
Directory	Является каталогом.
Encrypted	Зашифрован.
Hidden	Является скрытым.
Normal	Без специальных атрибутов.
Offline	Автономный.
ReadOnly	Доступный только для чтения.
System	Системный.
Temporary	Временный.

https://docs.microsoft.com/en-us/dotnet/api/system.io.fileattributes?view=net-5.0

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");
```

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).

Свойства Классов DirectoryInfo и FileInfo

```
FileAttributes Attributes {get; set;}
DateTime CreationTime {get; set;}
bool Exists {get;}
string FullName {get;}
DateTime LastAccessTime {get; set;}
DateTime LastWriteTime {get; set;}
string Name {get;}
DirectoryInfo Parent {get;}
DirectoryInfo Root {get;}
```

Статические Методы Класса File для Работы с Атрибутами:

```
FileAttributes GetAttributes(string path)
void SetAttributes(string path, FileAttributes fileAttributes)
```

Дополнительные Свойства FileInfo

```
DirectoryInfo Directory {get;}

string DirectoryName {get;}

string Extension {get;}

bool IsReadOnly {get; set;}

long Length {get;}

string Name {get;}
```

Дополнительные Методы DirectoryInfo

Аналоги методов из класса Directory:

- IEnumerable<DirectoryInfo> EnumerateDirectories()
- IEnumerable<FileInfo> EnumerateFiles()
- IEnumerable<FileSystemInfo> EnumerateFileSystemInfos()

- DirectoryInfo[ ] GetDirectories()
- FileInfo[ ] GetFiles()
- FileSystemInfo[ ] GetFileSystemInfos()

Дополнительные Методы FileInfo

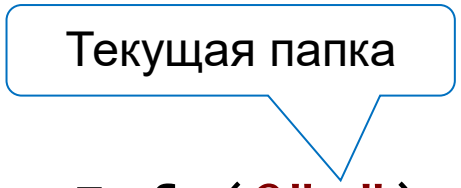
```
StreamWriter AppendText()
FileInfo CopyTo(string path)
StreamWriter CreateText()
FileStream Open(FileMode mode
                [, FileAccess access
                [, FileShare sharing_mode] ] )

FileStream OpenRead()
StreamReader OpenText()
FileStream OpenWrite()
FileInfo Replace(string path)
```

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}");
}
```



Результат Выполнения Программы

Вывод:

*** Имя каталога: net5.0
*** Имена и размеры файлов:
MyDirectory.deps.json; Length = 464
MyDirectory.dll; Length = 8192
MyDirectory.exe; Length = 125952
MyDirectory.pdb; Length = 10808
MyDirectory.runtimeconfig.dev.json; Length = 232
MyDirectory.runtimeconfig.json; Length = 147

Запись Средствами Класса File

```
void WriteAllText(string path, string data
                  [, System.Text.Encoding encoding]);

void WriteAllLines(string path, string[] data
                   [, System.Text.Encoding encoding]);

void WriteAllBytes(string path, byte[] data);
```

Чтение Средствами Класса File

```
string ReadAllText(string path
                   [, System.Text.Encoding encoding]);

string[] ReadAllLines(string path
                   [, System.Text.Encoding encoding]);

byte[] ReadAllBytes(string path);

IEnumerable<string> ReadLines(string path);
```

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

Кодирование Текстовой информации

Текст – последовательность символов алфавита.

Человек различает символы по их начертанию, а компьютер – по их двоичным кодам. При вводе в компьютер текстовой информации происходит её двоичное кодирование, изображение знака преобразуется в его двоичный код.

Исторически (однобайтовые кодировки):

Для представления текстовой информации достаточно 256 различных знаков.

Для кодирования **одного символа** требуется **один байт** информации.

https://en.wikipedia.org/wiki/Code_page

Таблица Кодировки

При кодировании каждому символу алфавита ставится в соответствие уникальный двоичный код.

Таблица кодировки – это таблица, в которой каждому символу алфавита сопоставляется соответствующий код.

ASCII (American Standard Code for Information Interchange) — стандартная кодировка изначально применённая на IBM PC (7 bit).

Коды:

- **от 0 до 32** соответствуют операциям (перевод строки, табуляция, пробела и т.д.);
- **от 33 по 127** соответствуют знакам латинского алфавита, цифрам, знакам арифметических операций и знакам препинания.

Кодировка ASCII

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F
0.	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1.	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2.		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3.	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4.	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5.	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6.	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7.	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Различные Кодировки

Коды от 128 по 255 – в различных национальных кодировках одному и тому же коду соответствуют разные символы.

Примеры различных таблиц кодировок для русских букв (Windows, MS-DOS, КОИ-8, Mac, ISO):

Символ	Windows	MS-DOS	КОИ-8	Mac	ISO	Unicode
А	192	128	225	128	176	1040
В	194	130	247	130	178	1042
М	204	140	237	140	188	1052
Э	221	157	252	157	205	1069
я	255	239	241	223	239	1103

Поэтому тексты, созданные в одной кодировке, не будут правильно отображаться в другой.

Кодировки в .NET

Пространство имён: **`System.Text`**

Библиотечный класс: **`Encoding`**

Статические свойства (часто используемые кодировки):

`ASCII, Default, Unicode, UTF8, UTF32`

Статические методы, доступные для кодировок:

```
virtual byte[] GetBytes(string s)
virtual string GetString(byte[] bytes)
virtual char[] GetChars(byte[] bytes, int index, int count)
virtual int GetCharCount(byte[] bytes)
```

Кодировки – Свойства Класса Encoding

Ниже перечислены часто используемые свойства класса System.Text.Encoding – наиболее часто встречающиеся варианты кодировок.

Имя	Описание
ASCII	Кодировка ASCII. Для представления символов используется 7 младших бит байта.
Default	Кодировка ANSI операционной системы. (Зависит от контекста, например, windows-1251).
Unicode	Кодировка UTF-16. Для представления символа латиницы используется 2 байта (… - до 4 байт).
UTF8	Кодировка UTF-8. Для представления символа латиницы используется 1 байт (кириллицы – 2 байта; … до 4 байт).

http://www.unicode.org/faq/utf_bom.html#UTF8

Кодировки. Получение Объекта-Кодировки

Метод, возвращающий кодировку, связанную с указанным номером кодовой страницы:

```
public static Encoding GetEncoding(int codePage);
```

Метод, возвращающий кодировку, связанную с указанным именем кодовой страницы:

```
public static Encoding GetEncoding(string name);
```

https://en.wikipedia.org/wiki/Code_page

Применение Средств Класса Encoding

```
using System.Text;

// 10 байтов.
byte[] utf8Bytes = Encoding.UTF8.GetBytes("0-нуль");
// 12 байтов.
byte[] utf16Bytes = Encoding.Unicode.GetBytes("0-нуль");

// строка «0-нуль».
string result1 = Encoding.UTF8.GetString(utf8Bytes);
string result2 = Encoding.Unicode.GetString(utf16Bytes);
```

Применение Методов Записи Текста

```
using System;
using System.IO;

string text = "two-два";
File.WriteAllText(@"c:\myText.txt", text); // UTF-8.
FileInfo fi = new FileInfo(@"c:\myText.txt");
Console.WriteLine("fi.Length = " + fi.Length);

string[] stringArray = { "two", "-", "два" };
File.WriteAllLines(@"c:\myLines.txt", stringArray); // UTF-8.
FileInfo fi2 = new FileInfo(@"c:\myLines.txt");
Console.WriteLine("fi2.Length = " + fi2.Length);
```

Вывод:
fi.Length = 10

Вывод:
fi2.Length = 16

Encoding и Запись Массива Байтов в Файл

```
using System;
using System.IO;        // File, FileInfo.
using System.Text;      // Encoding.

string text = "two-два";
byte[] byteArray = Encoding.Default.GetBytes(text); // UTF-8
Console.WriteLine("byteArray.Length = " + byteArray.Length);
File.WriteAllBytes(@"C:\myBytes.txt", byteArray);
FileInfo fi = new FileInfo(@"c:\myBytes.txt");
Console.WriteLine("fi.Length = " + fi.Length);
```

Вывод:
byteArray.Length = 10
fi.Length = 10

Запись Текста в Заданной Кодировке

```
using System;
using System.IO;
using System.Text;

string text = "two-два";
File.WriteAllText(@"c:\myText.txt", text, Encoding.UTF8);
byte[] arb = File.ReadAllBytes(@"c:\myText.txt");
Console.WriteLine("UTF8.Length = " + arb.Length);

File.WriteAllText(@"c:\myText.txt", text,Encoding.GetEncoding(866));
arb = File.ReadAllBytes(@"c:\myText.txt");
Console.WriteLine("MS-DOS.Length = " + arb.Length);
```

Пишется BOM

Вывод:
UTF8.Length = 13

Вывод:
MS-DOS.Length = 7

Запись и Чтение в Кодировке MS-DOS

```
using System;
using System.IO;
using System.Text;

string word = "concatenation - сцепление";
// Кодировка MS-DOS.
File.WriteAllText("msg.txt", word, Encoding.GetEncoding(866));
Console.WriteLine("Файл msg с кодировкой MS-DOS создан.");

string dosWord = File.ReadAllText("msg.txt");
Console.WriteLine("Текст из файла: \r\n" + dosWord);
// Кодировка MS-DOS.
string text = File.ReadAllText("msg.txt", Encoding.GetEncoding(866));
Console.WriteLine("Текст из файла: \r\n" + text);
```

Пример: Работа с Файлами и ООП. Часть 1

Warning: данный слайд выходит за рамки темы лекции и в первую очередь предназначен для более продвинутой аудитории.

Использование записей помогает значительно сократить код. Исчезает полная необходимость в реализации логики, связанной с объявлением и инициализацией полей.

```
record Student(string name, uint year, float mark)
{
    public static Student ReadStudent()
    {
        Console.Write("name = ");
        string name = Console.ReadLine();
        Console.Write("year = ");
        uint year = uint.Parse(Console.ReadLine());
        Console.Write("mark = ");
        float mark = float.Parse(Console.ReadLine());
        return new(name, year, mark);
    }
    // Student { name = <name>, year = <year>, mark = <mark> }
    public override string ToString() => $"{name}, {year}, {mark}";
}
```

ToString() переопределён для удобства конвертации объектов в строки!

Пример: Работа с Файлами и ООП. Часть 2

Warning: данный слайд выходит за рамки темы лекции и в первую очередь предназначен для более продвинутой аудитории.

```
class Project {
    static void Main(string[] args)
    {
        Console.Write("Количество студентов: ");
        int studentCount = int.Parse(Console.ReadLine());
        List<Student> studList = new(studentCount);
        //  Цикл ввода данных и добавления в список объектов класса Student:
        for (int i = 0; i < studentCount; ++i) {
            studList.Add(Student.ReadStudent());
        }

        string[] outList = Array.ConvertAll(studList.ToArray(),
                                            (Student s) => s.ToString());
        // Запись в файл массива строк:
        File.WriteAllLines(@"..\..\..\Group.txt", outList);
        MainRead();
    }
}
```

Класс в одном проекте с записью Student, осуществляет запись данных о студентах в файл и чтение из файла.

Конвертация каждого студента массива в строку путём вызова метода ToString().

Пример: Работа с Файлами и ООП. Часть 3

Warning: данный слайд выходит за рамки темы лекции и в первую очередь предназначен для более продвинутой аудитории.

```
class Project
{
    public static void MainRead()
    {
        string path = @"..\..\..\Group.txt";
        if (!File.Exists(path))
        {   // Завершение программы при отсутствии файла.
            return;
        }
        string[] inLines = File.ReadAllLines(path);
        Student[] resList = Array.ConvertAll(inLines, line => {
            string[] splitLines = line.Split(',');
            return new Student(splitLines[0],
                uint.Parse(splitLines[1]), float.Parse(splitLines[2]));
        });
        Array.ForEach(resList, student => Console.WriteLine(student));
    }
}
```

Конвертация строк обратно в объекты Student.

Применение метода Console.WriteLine() к каждому объекту Student.