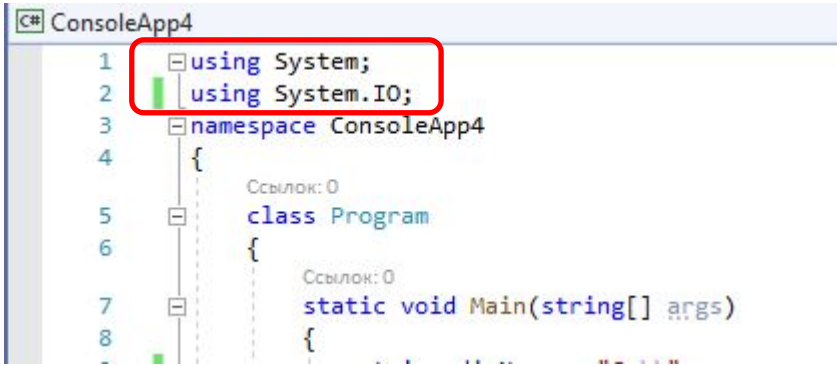
“Работа с файлами и каталогами”

Пространство имен **System.IO**

Фреймворк .NET предоставляет большие возможности по управлению и манипуляции файлами и каталогами, которые по большей части сосредоточены в пространстве имен **System.IO**. Классы, расположенные в этом пространстве имен (такие как Stream, StreamWriter, FileStream и др.), позволяют управлять файловым вводом-выводом.



Работа с дисками

Работа с дисками

Для представления диска в пространстве имен System.IO имеется класс DriveInfo.
Метод GetDrives возвращает имена всех логических дисков компьютера.
свойства:

- **AvailableFreeSpace**: указывает на объем доступного свободного места на диске в байтах
- **DriveFormat**: получает имя файловой системы
- **DriveType**: представляет тип диска
- **IsReady**: готов ли диск (например, DVD-диск может быть не вставлен в дисковод)
- **Name**: получает имя диска
- **TotalFreeSpace**: получает общий объем свободного места на диске в байтах
- **TotalSize**: общий размер диска в байтах
- **VolumeLabel**: получает или устанавливает метку тома

Работа с дисками

```
1  using System;
2  using System.IO;
3
4  namespace ConsoleApp3
5  {
       Ссылок: 0
6      class Program
7      {
           Ссылок: 0
8          static void Main(string[] args)
9          {
10             DriveInfo[] drives = DriveInfo.GetDrives();
11
12             foreach (DriveInfo drive in drives)
13             {
14                 Console.WriteLine($"Название: {drive.Name}");
15                 Console.WriteLine($"Тип: {drive.DriveType}");
16                 if (drive.IsReady)
17                 {
18                     Console.WriteLine($"Объем диска: {drive.TotalSize}");
19                     Console.WriteLine($"Свободное пространство: {drive.TotalFreeSpace}");
20                     Console.WriteLine($"Метка: {drive.VolumeLabel}");
21                 }
22                 Console.WriteLine();
23             }
24         }
25     }
26 }
```

Консоль отладки Microsoft Visual Studio

```
Название: C:\
Тип: Fixed
Объем диска: 119386664960
Свободное пространство: 45325807616
Метка:

Название: D:\
Тип: Fixed
Объем диска: 2000396742656
Свободное пространство: 461677674496
Метка: 18-3

Название: E:\
Тип: Fixed
Объем диска: 2000396742656
Свободное пространство: 86787805184
Метка: 17-11
```

Работа с каталогами

Работа с каталогами

Для работы с каталогами в пространстве имен System.IO предназначены два класса: **Directory** и **DirectoryInfo**.

- Класс Directory объявлен и определен в корне System.IO. Содержит в себе группу статических членов для создания, удаления, перемещения и переименования каталогов и подкаталогов.
  Запечатанный (sealed - предотвращает наследование) и статический (static) тип: не может иметь наследников, объект класса создать нельзя.
- Класс DirectoryInfo - ссылочный тип, наследуется от System.IO.FileSystemInfo, поэтому имеет множество перегруженных методов родительского класса, а также некоторые полезные свойства при работе с каталогами.

Класс Directory

Класс Directory предоставляет ряд статических методов для управления каталогами. Некоторые из этих методов:

- **CreateDirectory(path):** создает каталог по указанному пути path
- **Delete(path):** удаляет каталог по указанному пути path
- **Exists(path):** определяет, существует ли каталог по указанному пути path. Если существует, возвращается true, если не существует, то false
- **GetDirectories(path):** получает список каталогов в каталоге path
- **GetFiles(path):** получает список файлов в каталоге path
- **Move(sourceDirName, destDirName):** перемещает каталог
- **GetParent(path):** получение родительского каталога

Класс Directory

Обратите внимание на использование слешей в именах файлов. Либо используется двойной слеш: "C:\\", либо одинарный, но тогда перед всем путем ставится знак @: @"C:\Program Files"

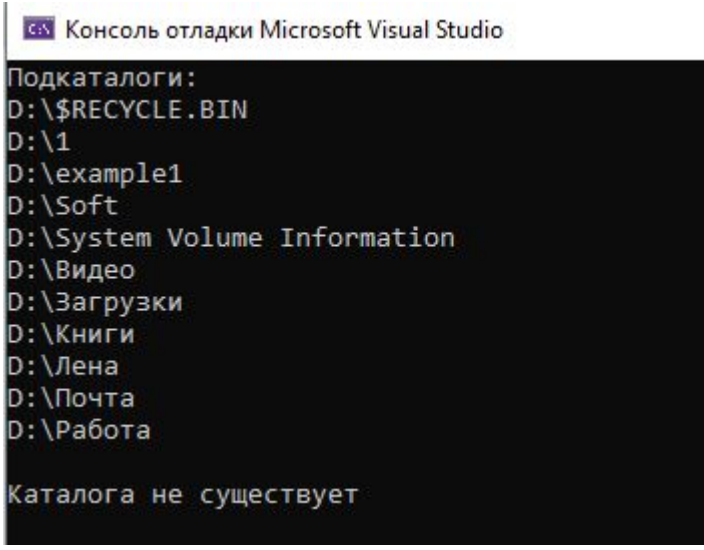
```
using System;
using System.IO;
namespace ConsoleApp4
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            string dirName = "C:\\";

            if (Directory.Exists(dirName))
            {
                Console.WriteLine("Подкаталоги:");
                string[] dirs = Directory.GetDirectories(dirName);
                foreach (string s in dirs)
                {
                    Console.WriteLine(s);
                }
                Console.WriteLine();
                Console.WriteLine("Файлы:");
                string[] files = Directory.GetFiles(dirName);
                foreach (string s in files)
                {
                    Console.WriteLine(s);
                }
            }
        }
    }
}
```

Пример получения списка файлов и

```
Консоль отладки Microsoft Visual Studio
Подкаталоги:
C:\$RECYCLE.BIN
C:\$WINDOWS.~BT
C:\Documents and Settings
C:\install
C:\Live! Cam
C:\LJP1100_P1560_P1600_Full_Solution
C:\Program Files
C:\Program Files (x86)
C:\ProgramData
C:\Recovery
C:\System Volume Information
C:\TCPU70
C:\Users
C:\Windows

Файлы:
C:\$WINRE_BACKUP_PARTITION.MARKER
C:\pagefile.sys
C:\swapfile.sys
```

Создание/удаление каталога

```
string dirName = "D:\\";
string dirName1 = @"D:\example1\example2";

if (Directory.Exists(dirName))
{
    Console.WriteLine("Подкаталоги:");
    string[] dirs = Directory.GetDirectories(dirName);
    foreach (string s in dirs)
    {
        Console.WriteLine(s);
    }
    Console.WriteLine();

}
Directory.CreateDirectory(@"D:\example1\example3");
if (Directory.Exists(dirName1))
{
    Directory.Delete(dirName1);

}
else Console.WriteLine("Каталога не существует");
```



блок try\catch

Поскольку каждое использование функционала класса Directory сопряжено либо с проверкой введенных пользователем путями, либо с файлами внутри каталога рекомендуется включать его в блок try\catch. Если путь будет задан неправильно, это вызовет исключение.

```
using System;
using System.IO;

namespace ConsoleApp3
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            try
            {
                string dirName = "D:\\";
                string dirName1 = @"D:\example1\example2";

                if (Directory.Exists(dirName))
                {
                    Console.WriteLine("Подкаталоги:");
                    string[] dirs = Directory.GetDirectories(dirName);
                    foreach (string s in dirs)
                    {
                        Console.WriteLine(s);
                    }
                    Console.WriteLine();
                    Console.WriteLine("Файлы:");
                    string[] files = Directory.GetFiles(dirName);
                    foreach (string s in files)
                    {
                        Console.WriteLine(s);
                    }
                }
                Directory.CreateDirectory(@"D:\example1\example3");
               // if (Directory.Exists(dirName1))
                //{
                    Directory.Delete(dirName1);

                // }
            }
            catch (Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
    }
}
```

Консоль отладки Microsoft Visual Studio

```
D:\$RECYCLE.BIN
D:\1
D:\example1
D:\Soft
D:\System Volume Information
D:\Видео
D:\Загрузки
D:\Книги
D:\Лена
D:\Почта
D:\Работа

Файлы:
D:\17-11 (F) - Ярлык.lnk
D:\EXP_INFO_1.data
Could not find a part of the path 'D:\example1\example2'.
```

Класс DirectoryInfo

Данный класс предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на Directory.

Некоторые из его свойств и методов:

- **Create()**: создает каталог
- **CreateSubdirectory(path)**: создает подкаталог по указанному пути path
- **Delete()**: удаляет каталог
- Свойство **Exists**: определяет, существует ли каталог
- **GetDirectories()**: получает список каталогов
- **GetFiles()**: получает список файлов
- **MoveTo(destDirName)**: перемещает каталог
- Свойство **Parent**: получение родительского каталога
- Свойство **Root**: получение корневого каталога

Класс DirectoryInfo

Поскольку **DirectoryInfo** – ссылочный тип(класс), прежде чем воспользоваться его членами нужно создать его объект с помощью конструктора. В качестве параметра конструктора необходимо передать строку с путем и именем каталога. Каталог будет назначен текущим рабочим каталогом для созданного экземпляра DirectInfo.

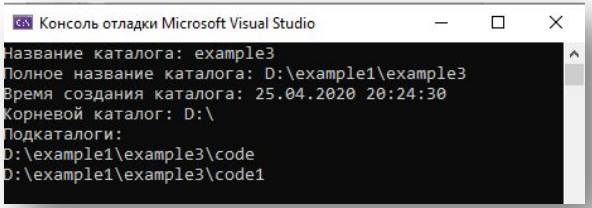
```
1   using System;
2   using System.IO;
3
4   namespace ConsoleApp1
5   {
        Ссылок: 0
6       class Program
7       {
            Ссылок: 0
8           static void Main(string[] args)
9           {
10              string dirName = @"D:\example1\example3";
11
12              DirectoryInfo dirInfo = new DirectoryInfo(dirName);
13
14              Console.WriteLine($"Название каталога: {dirInfo.Name}");
15              Console.WriteLine($"Полное название каталога: {dirInfo.FullName}");
16              Console.WriteLine($"Время создания каталога: {dirInfo.CreationTime}");
17              Console.WriteLine($"Корневой каталог: {dirInfo.Root}");
18          }
19      }
20  }
21
```

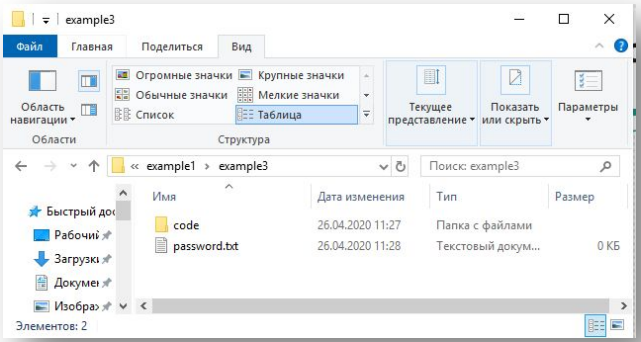
```
Название каталога: example3
Полное название каталога: D:\example1\example3
Время создания каталога: 25.04.2020 20:24:30
Корневой каталог: D:\
```

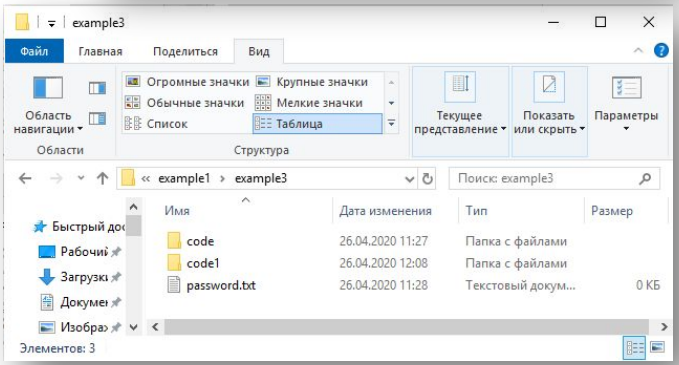
Создание каталога

```
{
    string dirName = @"D:\example1\example3";

    DirectoryInfo dirInfo = new DirectoryInfo(dirName);

    string path = @"D:\example1";
    string subpath = @"example3\code1";
    DirectoryInfo dirInfoA = new DirectoryInfo(path);
    if (!dirInfo.Exists)
    {
        dirInfoA.Create();
    }
    dirInfoA.CreateSubdirectory(subpath);

    Console.WriteLine("Подкаталоги:");
    string[] dirs = Directory.GetDirectories(dirName);
    foreach (string s in dirs)
    {
        Console.WriteLine(s);
    }
}
```

Сначала проверяем, нет ли такой директории, так как если она существует, то ее создать будет нельзя, и приложение выбросит ошибку. В итоге у нас получится следующий путь: "D:\example1\example3\code1"







Работа с файлами

Работа с файлами

- **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.

  Под работой с файлами подразумевается:
  - создание файлов;
  - удаление файлов;
  - чтение данных;
  - запись данных;
  - изменение параметров файла (имя, расширение...);
  - другое.

Классы File и FileInfo

Для работы с файлами в пространстве имен System.IO предназначены два класса: **File** и **FileInfo**
Различие между классами кроется в механизме взаимодействия с ними.

- Все члены класса **File** – статические, поэтому он позволяет производить файловые операции **без необходимости создавать объект типа**. Все его методы требуют указания файлового пути, то есть адресной ссылки внутри файловой системы.
  Статичность методов класса File делает его использование предпочтительным в большинстве случаев, когда файловая операция – одиночная.
- Когда существует необходимость множественных действий с конкретным файлом, то безопасней и удобней создать объектную ссылку. Такой подход заключен в классе FileInfo. Функционал этого класса заметно шире.

Класс FileInfo

Предоставляет свойства и методы экземпляра для создания, копирования, удаления, перемещения и открытия файлов

Основные методы и свойства:

- CopyTo(path): копирует файл в новое место по указанному пути path
- Create(): создает файл
- Delete(): удаляет файл
- MoveTo(destFileName): перемещает файл в новое место
- Свойство Exists: указывает, существует ли файл
- Свойство Length: получает размер файла
- Свойство Name: получает имя файла

Абсолютный и относительный путь

Абсолютные пути полностью указывают расположение: файл или каталог могут быть однозначно идентифицированы независимо от текущего расположения.

```
string absolutePath = @"c:\temp\MyTest.txt"; //абсолютный
```

Относительные пути указывают на неполное расположение: текущее расположение используется в качестве отправной точки при поиске файла, указанного относительного пути.

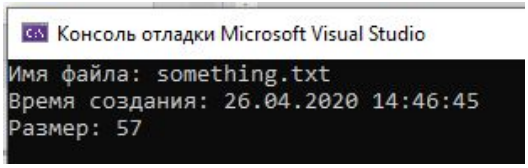
```
string relativePath = "MyTest.txt"; //относительный
```

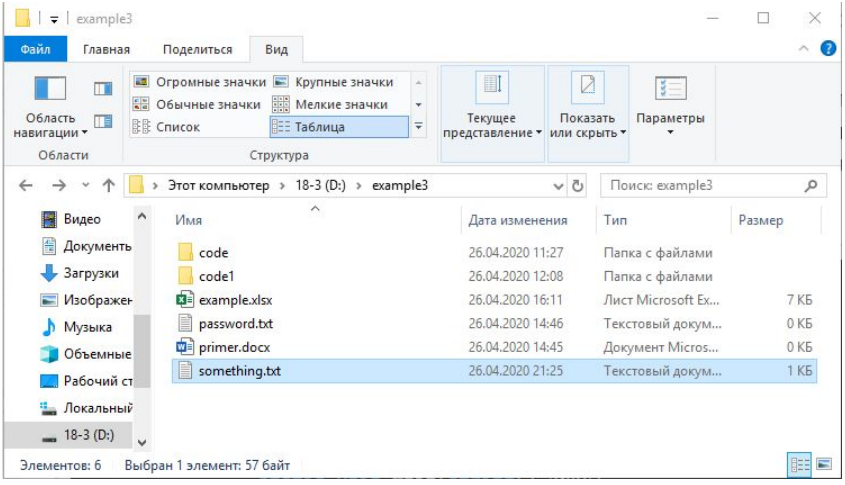
Примеры правильной строки:

```
«c:\\SomeDir\\SomeFile.dat» - строка с указанием конечного файла.
«c:\\SomeDir» - строка с указанием на директорию с файлом.
«c:\\SomeDir\\SomeSubDir» - строка с указанием на поддиректорий с фалом.
```

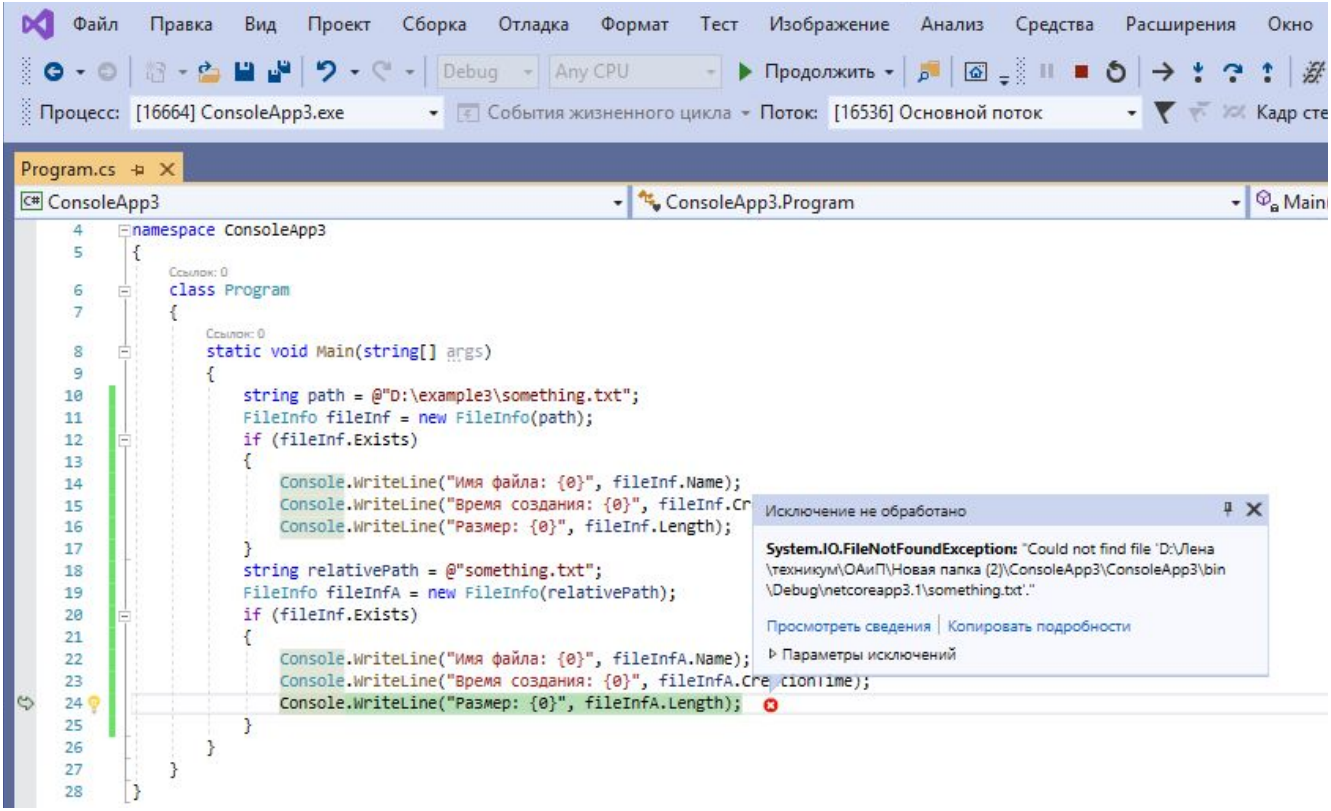
Получение информации о файле

```
1  using System;
2  using System.IO;
3
4  namespace ConsoleApp3
5  {
       Ссылок: 0
6      class Program
7      {
           Ссылок: 0
8          static void Main(string[] args)
9          {
10             string path = @"D:\example3\something.txt";
11             FileInfo fileInf = new FileInfo(path);
12             if (fileInf.Exists)
13             {
14                 Console.WriteLine("Имя файла: {0}", fileInf.Name);
15                 Console.WriteLine("Время создания: {0}", fileInf.CreationTime);
16                 Console.WriteLine("Размер: {0}", fileInf.Length);
17             }
18         }
19     }
20 }
21
```





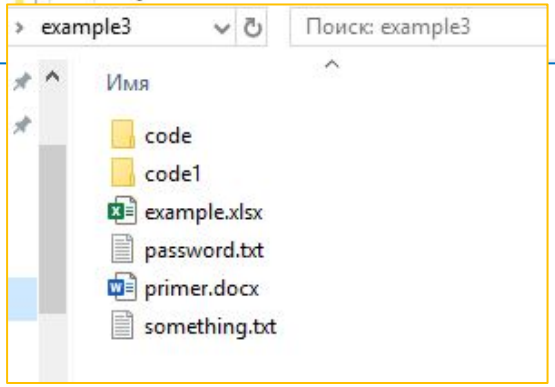
Получение информации о файле (пример с неверным путем)

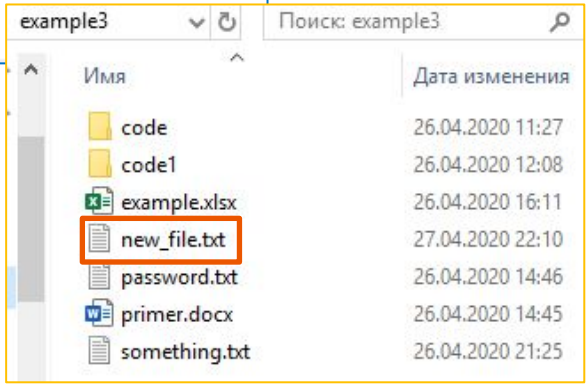


Создание пустого файла (абсолютный путь)

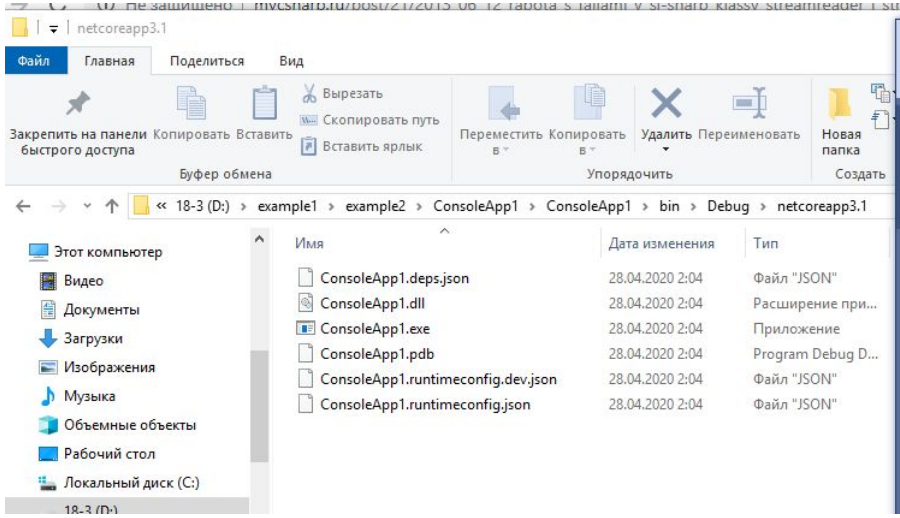
```
using System;
using System.IO;

namespace ConsoleApp3
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            string path = @"D:\example3\something.txt";
            FileInfo fileInf = new FileInfo(path);
            if (fileInf.Exists)
            {
                Console.WriteLine("Имя файла: {0}", fileInf.Name);
                Console.WriteLine("Время создания: {0}", fileInf.CreationTime);
                Console.WriteLine("Размер: {0}", fileInf.Length);
            }
            File.Create("D:\\example3\\new_file.txt");
        }
```

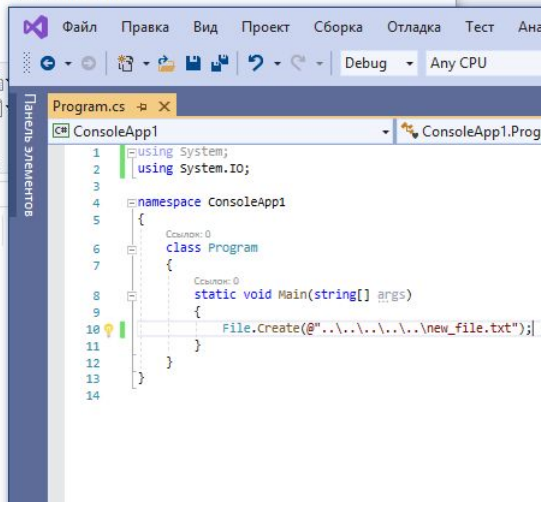
Если файл с таким именем уже существует, он будет переписан на новый пустой файл.

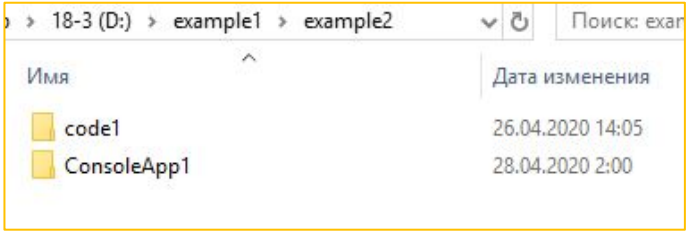


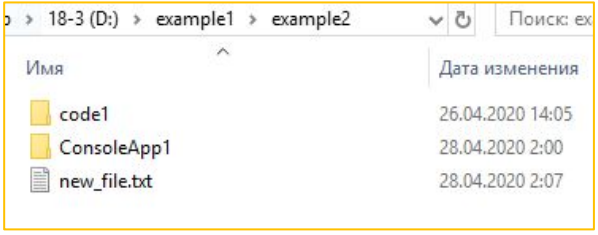


Создание пустого файла (относительный путь)









Чтение данных и запись данных в файл

Способы чтения и записи в файл:

- Статическими методами класса File.
- С помощью методов классов StreamReader и Streamwriter.
- С помощью методов классов BinaryReader и BinaryWriter.

Первый способ самый удобный, поскольку позволяет считать или заполнить весь файл буквально в одну строку кода. Благодаря тому, что методы класса **File** статические, нет необходимости создавать экземпляры объектов, работающих с файлами. Для большинства задач его методы подходят.

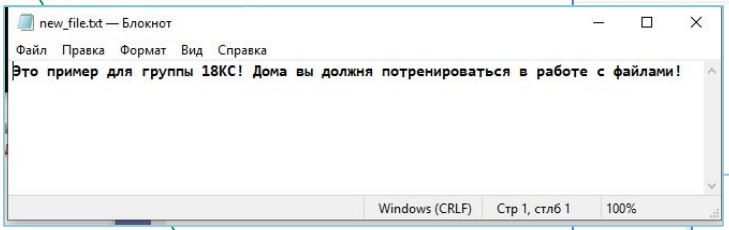
Объекты **StreamReader** и **StreamWriter** (**BinaryReader** и **BinaryWriter**) возвращаются методами класса **FileInfo**, как экземпляры, непосредственно работающие с файлом. Код становится более громоздким, но и более гибким, поскольку позволяет организовать сложную систему считывания/записи данных файл, внутри единого блока try/catch для обработки исключений.

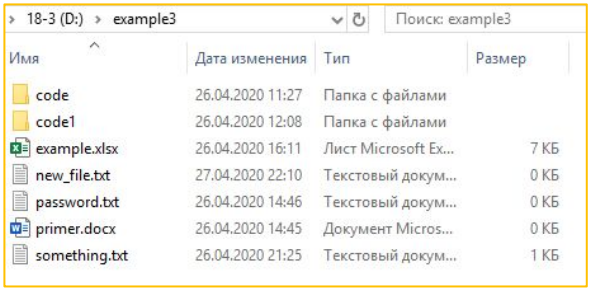
Чтение/запись статическими методами класса File.

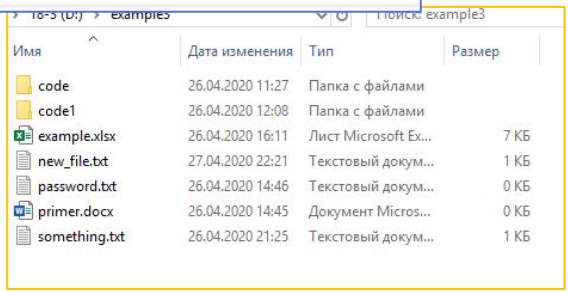
Запись текста в файл

Метод **WriteAllText()** создает новый файл (если такого нет), либо открывает существующий и записывает текст, заменяя всё, что было в файле. Метод имеет два параметра: строковое имя файла и строку или массив строк, которые нужно записать в файл. Метод сам откроет файл, запишет в него текст и закроет его.

```
using System;
using System.IO;

namespace ConsoleApp4
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            File.WriteAllText("D:\\example3\\new_file.txt", "Это пример для группы 18КС! " +
                "Дома вы должня потренироваться в работе с файлами!");
        }
    }
}
```



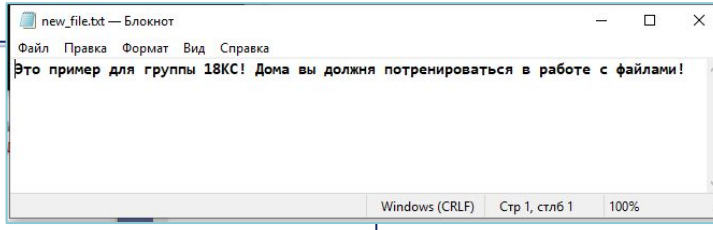


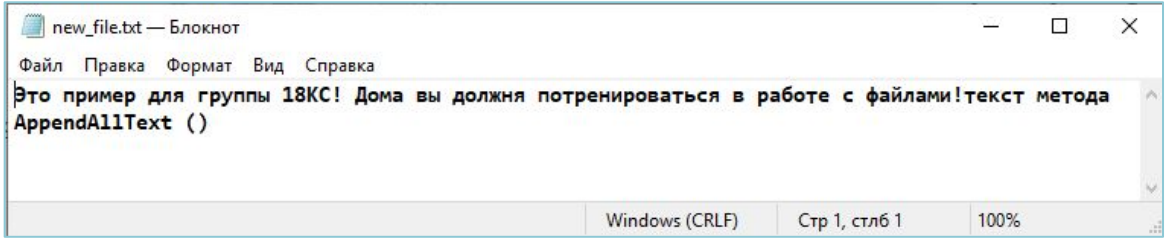


Запись текста в файл

Метод **AppendAllText()** работает, как и метод WriteAllText() за исключением того, что новый текст дописывается в конец файла, а не переписывает всё что было в файле.

```
using System;
using System.IO;

namespace ConsoleApp4
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            File.WriteAllText("D:\\example3\\new_file.txt", "Это пример для группы 18КС! " +
                "Дома вы должня потренироваться в работе с файлами!");
            File.AppendAllText("D:\\example3\\new_file.txt", "текст метода AppendAllText ()");
        }
    }
}
```





Считывание данных из файла

Функционал класса **File** позволяет считывать из файла данные в виде строк. Основной метод, который используется при этом – **ReadAllText()**.

При вызове через имя класса он:

- Открывает файл.
- Считывает все строки файла.
- Закрывает файл.

Считывание данных из файла

Функционал класса **File** позволяет считывать из файла данные в виде строк. Основной метод, который используется при этом – **ReadAllText()**.

```
using System;
using System.IO;


namespace ConsoleApp4
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            File.WriteAllText("D:\\example3\\new_file.txt", "Это пример для группы 18КС! " +
                "Дома вы должня потренироваться в работе с файлами!");
            File.AppendAllText("D:\\example3\\new_file.txt", "текст метода AppendAllText ()");

            string Filepath = "D:\\example3\\new_file.txt";
            if (File.Exists(Filepath))
            {
                string readText = File.ReadAllText(Filepath);
                Console.WriteLine("В файле {0} содержится: \n", Filepath);
                Console.WriteLine(readText);
            }
        }
    }
}
```

```
Консоль отладки Microsoft Visual Studio
В файле D:\example3\new_file.txt содержится:

Это пример для группы 18КС! Дома вы должня потренироваться в работе с файлами!текст метода AppendAllText ()
```

Чтение/запись с помощью методов классов StreamReader и Streamwriter.

Поток

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 Си-шарп.

Для работы с файлами используется класс **FileStream** (файловый поток).

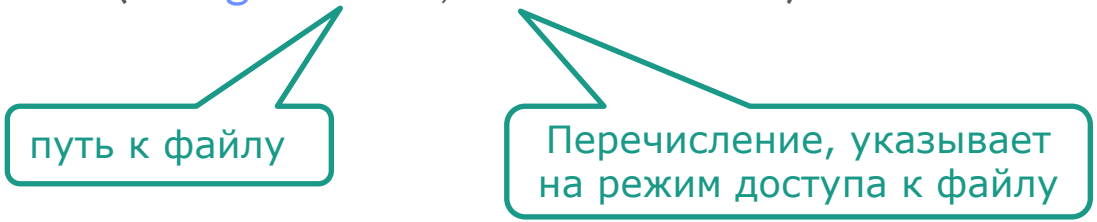
FileStream - представляет поток, который позволяет выполнять операции чтения/записи в файл.

Создание FileStream

Для создания объекта FileStream можно использовать как конструкторы этого класса, так и статические методы класса File.

Один из конструкторов имеет вид:

FileStream(string filename, FileMode mode)



Создание FileStream

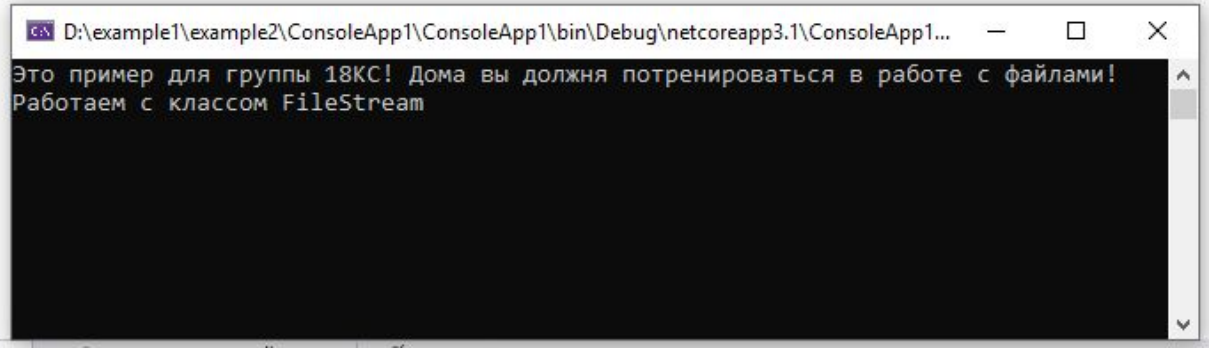
Режимы доступа к файлу:

- **Append**: если файл существует, то текст добавляется в конец файл. Если файла нет, то он создается. Файл открывается только для записи.
- **Create**: создается новый файл. Если такой файл уже существует, то он перезаписывается
- **CreateNew**: создается новый файл. Если такой файл уже существует, то он приложение выбрасывает ошибку
- **Open**: открывает файл. Если файл не существует, выбрасывается исключение.
- **OpenOrCreate**: если файл существует, он открывается, если нет - создается новый
- **Truncate**: если файл существует, то он перезаписывается. Файл открывается только для записи.

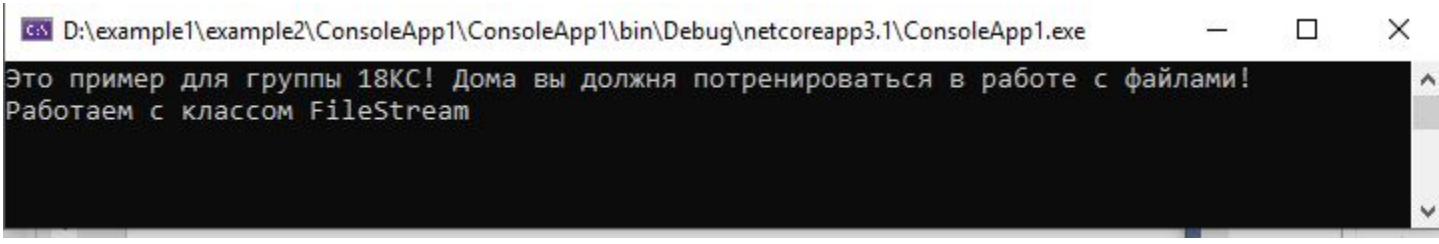
Чтение из файла

Для чтения данных из потока используется класс **StreamReader**

```
1  using System;
2  using System.IO;
3
4  namespace ConsoleApp1
5  {
       Ссылок: 0
6      class Program
7      {
           Ссылок: 0
8          static void Main(string[] args)
9          {
10             FileStream file1 = new FileStream("d:\\example3\\new_file.txt", FileMode.Open); //создаем файловый поток
11             StreamReader reader = new StreamReader(file1); // создаем «потоковый читатель» и связываем его с файловым потоком
12             Console.WriteLine(reader.ReadToEnd()); //считываем все данные с потока и выводим на экран
13             reader.Close(); //закрываем поток
14             Console.ReadLine();
15         }
16     }
17 }
18
```



```
1  using System;
2  using System.IO;
3
4  namespace ConsoleApp1
5  {
       Ссылок: 0
6      class Program
7      {
           Ссылок: 0
8          static void Main(string[] args)
9          {
10             FileStream file1 = new FileStream("d:\\example3\\new_file.txt", FileMode.Open); //создаем файловый поток
11             StreamReader reader = new StreamReader(file1); // создаем «потоковый читатель» и связываем его с файловым потоком
12             Console.WriteLine(reader.ReadToEnd()); //считываем все данные с потока и выводим на экран
13             reader.Close(); //закрываем поток
14             Console.ReadLine();
15
16
17         }
18     }
19 }
```



Запись в файл

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

```
using System;
using System.IO;

namespace ConsoleApp1
{
    Ссылок: 0
    class Program
    {
        Ссылок: 0
        static void Main(string[] args)
        {
            FileStream file2 = new FileStream(@"D:\example3\password.txt", FileMode.Create); //создаем файловый поток
            StreamWriter writer = new StreamWriter(file2); //создаем «потоковый писатель» и связываем его с файловым потоком
            writer.Write("!!!Добавленный текст!!!"); //записываем в файл
            writer.Close(); //закрываем поток. Не закрыв поток, в файл ничего не запишется


        }
    }
}
```