Работа с файловой системой

- В пространстве имен **System.IO** предусмотрено четыре класса, которые предназначены для работы с файловой системой компьютера.

- ***Directory*** и ***Fi1e*** реализуют свои возможности с помощью статических методов, поэтому данные классы можно использовать без создания соответствующих объектов (экземпляров классов).

- ***DirectoryInfo*** и ***FileInfo*** обладают схожими функциональными возможностями с Directory и Fi1e, но порождены от класса FileSystemInfo и поэтому реализуются путем создания соответствующих экземпляров классов.

Работа с каталогами
Абстрактный класс FileSystemInfo

В FileSystemInfo предусмотрено несколько методов:

- метод Delete() - позволяет удалить объект файловой системы с жесткого диска
- Refresh() — обновить информацию об объекте файловой системы.

Некоторые свойства FileSystemInfo

Свойство	*Описание*
Attributes	Позволяет получить или установить атрибуты для данного объекта файловой системы. Для этого свойства используются значения и перечисления FileAttributes
CreationTime	Позволяет получить или установить время создания объекта файловой системы
Exists	Может быть использовано для того, чтобы определить, существует ли данный объект файловой системы
Extension	Позволяет получить расширение для файла
FullName	Возвращает имя файла или каталога с указанием пути к нему в файловой системе
LastAccessTime	Позволяет получить или установить время последнего обращения к объекту файловой системы
LastWriteTime	Позволяет получить или установить время последнего внесения изменений в объект файловой системы
Name	Возвращает имя указанного файла. Доступно только для чтения. Для каталогов возвращает имя последнего каталога в иерархии, если это возможно. Если нет, возвращает полностью определенное имя

Класс DirectoryInfo

- Наследует члены класса FileSystemInfo и содержит дополнительный набор членов, которые предназначены для :
- создания,
- перемещения,
- удаления,
- получения информации о каталогах и подкаталогах в файловой системе.

Класс DirectoryInfo

Член	Описание
Create() **CreateSubDirectory()**	Создают каталог (или подкаталог) по указанному пути в файловой системе
Delete()	Удаляет пустой каталог
GetDirectories()	Позволяет получить доступ к подкаталогам текущего каталога (в виде массива объектов DirectoryInfo)
GetFiles()	Позволяет получить доступ к файлам текущего каталога (в виде массива объектов FileInfo)
MoveTo()	Перемещает каталог и все его содержимое на новый адрес в файловой системе
Parent	Возвращает родительский каталог в иерархии файловой системы

Тип DirectoryInfo

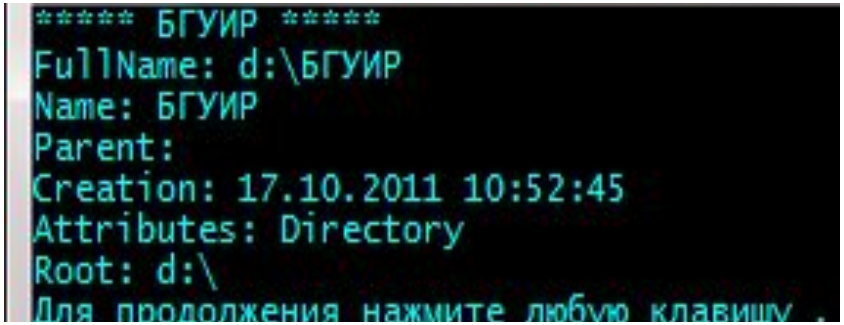
- создаем экземпляр класса (объект), указывая при вызове конструктора в качестве параметра путь к нужному каталогу

```
/*Создаем объект DirectoryInfo, которому будет обращаться к текущему каталогу*/
    DirectoryInfo dir1 = new DirectoryInfo(".");
    /*Создаем объект DirectoryInfo, которому будет обращаться к каталогу d:\prim*/
    DirectoryInfo dir2 = new DirectoryInfo(@"d:\prim");
```

- Если создается объект ***DirectoryInfo*** и связывается с несуществующим каталогом, то будет сгенерировано исключение ***System.IO.DirectoryNotFoundException***.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
namespace ConsoleApplication1
{
//создаем объект DIrectoryInfo, который связан с каталогом d:\БГУИР,
    выводим информацию о каталоге

class Program {
static void Main(string[] args) {
DirectoryInfo dir = new DirectoryInfo(@"d:\БГУИР");
Console.WriteLine("***** "+dir.Name+" *****");
Console.WriteLine("FullName: {0}", dir.FullName);
Console.WriteLine("Name: {0}", dir.Name);
Console.WriteLine("Parent: {0}", dir.Parent);
Console.WriteLine("Creation: {0}", dir.CreationTime);
Console.WriteLine("Attributes: {0}", dir.Attributes.ToString());
Console.WriteLine("Root: {0}", dir.Root);
}   }}     }
```



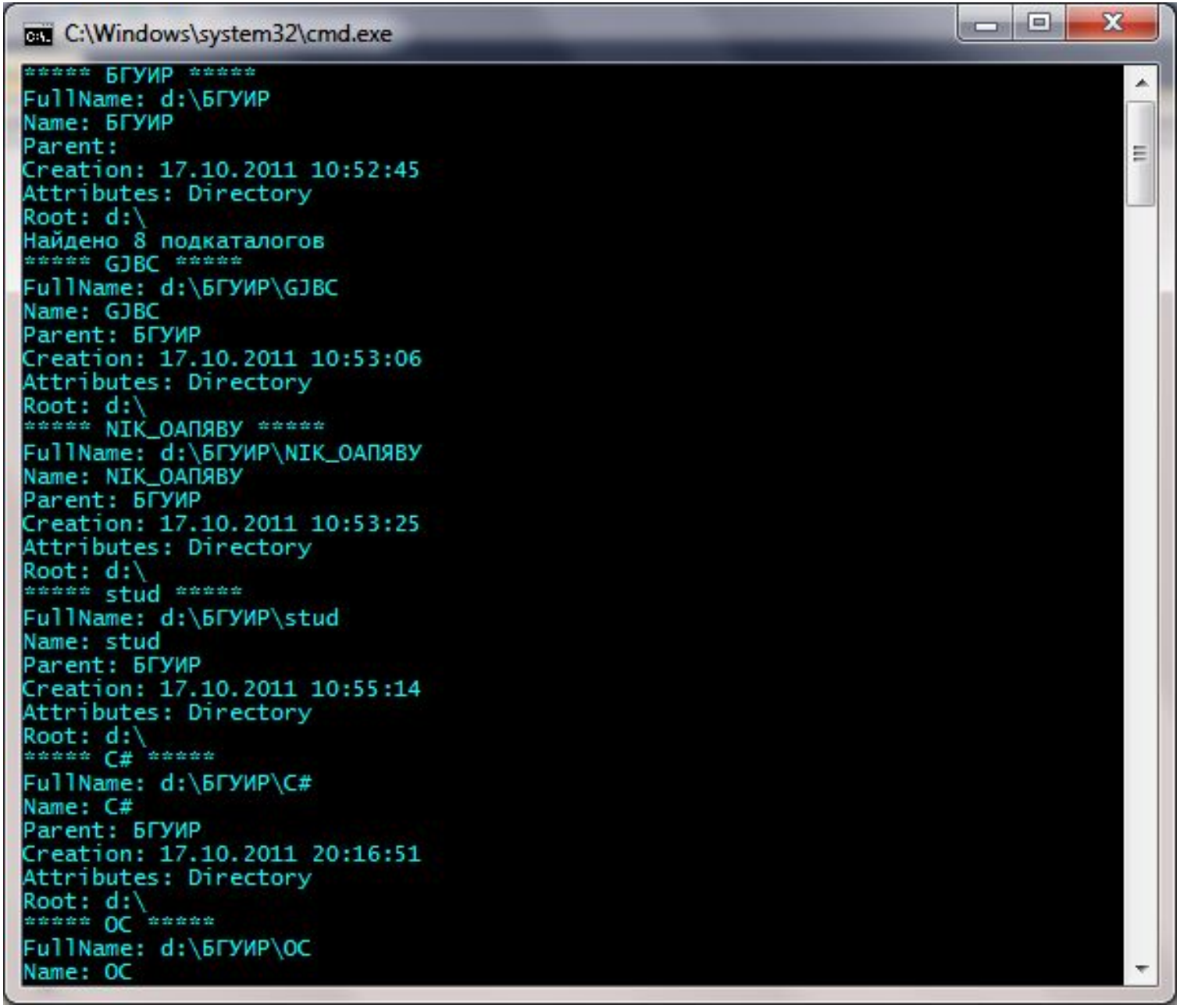
Свойство Attributes позволяет получить информацию об атрибутах объекта файловой системы

Значение	Описание
Archive	Этот атрибут используется приложениями при проведении резервного копирования, а в некоторых случаях — удаления старых файлов
Compressed	Определяет, что файл является сжатым
Directory	Определяет, что объект файловой системы является каталогом
Encrypted	Определяет, что файл является зашифрованным
Hidden	Определяет, что файл является скрытым (такой файл не будет выводиться при обычном просмотре каталога)
Normal	Определяет, что файл находится в обычном состоянии и для него установлены любые другие атрибуты. Этот атрибут не может использоваться с другими атрибутами
Offline	Файл (расположенный на сервере) кэширован в хранилище off-line на клиентском компьютере. Возможно, что данные этого файла уже устарели
Readonly	Файл доступен только для чтения
System	Файл является системным (то есть файл является частью операционной системы или используется исключительно операционной системой)

//Через DirectoryInfo можно не только получать доступ к информации о текущем каталоге, но получить доступ к информации о его подкаталогах

```
class Program       {
   static void printDirect(DirectoryInfo dir)          {
      Console.WriteLine("***** " + dir.Name + " *****");
      Console.WriteLine("FullName: {0}", dir.FullName);
      Console.WriteLine("Name: {0}", dir.Name);
      Console.WriteLine("Parent: {0}", dir.Parent);
      Console.WriteLine("Creation: {0}", dir.CreationTime);
      Console.WriteLine("Attributes: {0}", dir.Attributes.ToString());
      Console.WriteLine("Root: {0}", dir.Root);
   }

   static void Main(string[] args)          {
      DirectoryInfo dir = new DirectoryInfo(@"d:\БГУИР");
      printDirect(dir);
      DirectoryInfo[] subDirects = dir.GetDirectories();
      Console.WriteLine("Найдено {0} подкаталогов", subDirects.Length);
      foreach (DirectoryInfo d in subDirects)
      {                printDirect(d);             }
   }
}
```


C:\Windows\system32\cmd.exe
***** БГУИР *****
FullName: d:\БГУИР
Name: БГУИР
Parent:
Creation: 17.10.2011 10:52:45
Attributes: Directory
Root: d:\
Найдено 8 подкаталогов
***** GJBC *****
FullName: d:\БГУИР\GJBC
Name: GJBC
Parent: БГУИР
Creation: 17.10.2011 10:53:06
Attributes: Directory
Root: d:\
***** NIK_OAПЯВУ *****
FullName: d:\БГУИР\NIK_OAПЯВУ
Name: NIK_OAПЯВУ
Parent: БГУИР
Creation: 17.10.2011 10:53:25
Attributes: Directory
Root: d:\
***** stud *****
FullName: d:\БГУИР\stud
Name: stud
Parent: БГУИР
Creation: 17.10.2011 10:55:14
Attributes: Directory
Root: d:\
***** C# *****
FullName: d:\БГУИР\C#
Name: C#
Parent: БГУИР
Creation: 17.10.2011 20:16:51
Attributes: Directory
Root: d:\
***** ОС *****
FullName: d:\БГУИР\ОС
Name: ОС

Метод CreateSubdirectory()

- позволяет создать в выбранном каталоге как единственный подкаталог, так и множество подкаталогов (в том числе, и вложенных друг в друга).

Создадим в каталоге несколько дополнительных подкаталогов:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\БГУИР");
dir.CreateSubdirectory("doc"); //создали подкаталог
dir.CreateSubdirectory(@"book\2012");   //создали
    вложенный подкаталог
```

Метод MoveTo()

позволяет переместить текущий каталог по заданному в качестве параметра адресу. При этом возможно произвести переименование каталога.

```
DirectoryInfo dir = new DirectoryInfo( @"d:\БГУИР\bmp");
dir.( @"d:\БГУИР\ggg\bmp");
```

В данном случае каталог bmp перемещается в по адресу d:\БГУИР\ggg\bmp
Так как имя перемещаемого каталога совпадает с крайним правым именем в адресе нового местоположения каталога, то переименования не происходит.

//переименование текущего каталога:

```
DirectoryInfo dir = new DirectoryInfo(@"d:\БГУИР\ggg");
dir.MoveTo(@"d:\БГУИР\archive");
```

Замечания.

1. Удаление каталога возможно только тогда, когда он пуст.
2. На практике комбинируют использование классов Directory и DirectoryInfo.

Работа с файлами
Класс FileInfo

- предназначен для организации доступа к физическому файлу, который содержится на жестком диске компьютера.

- позволяет получать информацию об этом файле (о времени его создания, размере, атрибутах и т. п.),

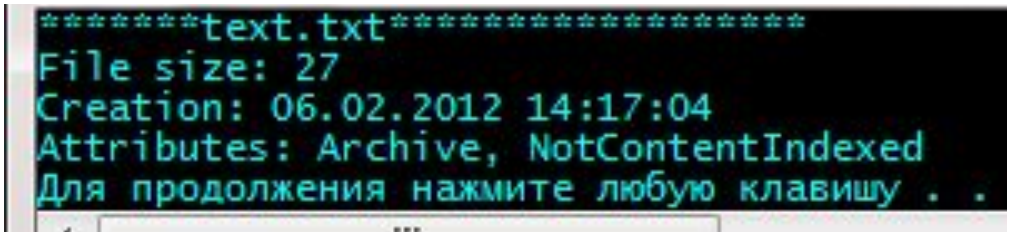
- производить различные операции (по созданию файла или его удалению).

- наследует члены класса FileSystemInfo и содержит дополнительный набор членов

Класс FileInfo наследует члены класса FileSystemInfo и содержит дополнительный набор членов

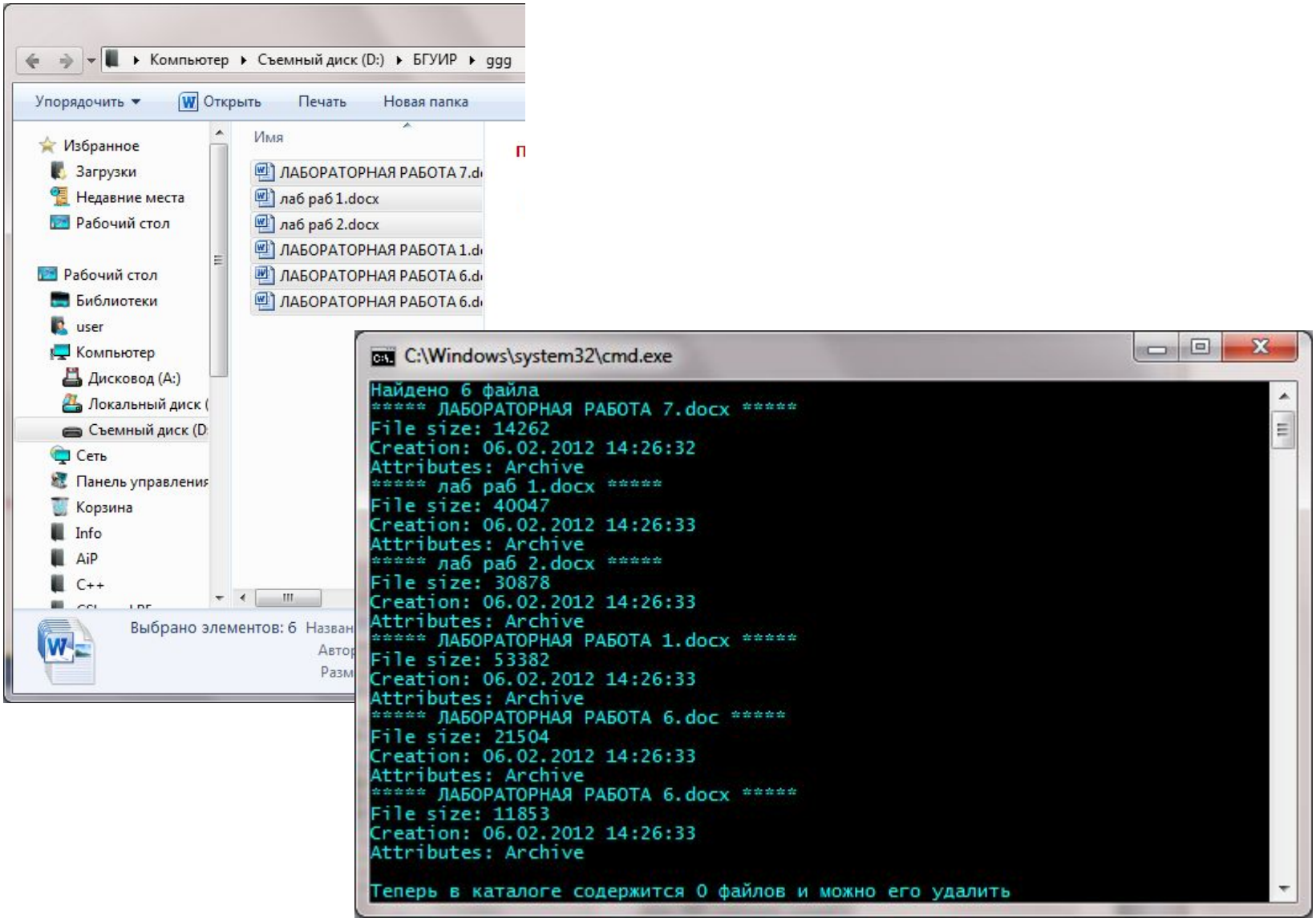
Член	*Описание*
AppendText()	Создает объект StreamWriter для добавления текста к файлу
CopyTo()	Копирует уже существующий файл в новый файл
Create()	Создает новый файл и возвращает объект FileStream для взаимодействия с этим файлом
CreateText()	Создает объект StreamWriter для записи текстовых данных в новый файл
Delete()	Удаляет файл, которому соответствует объект FileInfo
Directory	Возвращает каталог, в котором расположен данный файл
DirectoryName	Возвращает полный путь к данному файлу в файловой системе
Length	Возвращает размер файла
MoveTo()	Перемещает файл в указанное пользователем место (этот метод позволяет одновременно переименовать данный файл)
Name	Позволяет получить имя файла
Open()	Открывает файл с указанными пользователем правами доступа на чтение, запись или совместное использование с другими пользователями
OpenRead()	Создает объект FileStream, доступный только для чтения
OpenText()	Создает объект StreamReader, который позволяет считывать информацию из существующего текстового файла
OpenWrite()	Создает объект FileStream, доступный для чтения и записи

```
class Program       {
          static void Main(){
//создаем новый файл  и связываем с ним строковый поток
FileInfo f = new FileInfo("text.txt");
StreamWriter fOut = new StreamWriter(f.Create());
//записываем в файл данные и закрываем строковый поток,
// при этом  связь с физическим файлом для f не рвется
fOut.WriteLine("ОДИН ДВА ТРИ...");
fOut.Close();
//получаем информацию о файле
Console.WriteLine("*******" + f.Name+ "********************");
Console.WriteLine("File size: {0}", f.Length);
Console.WriteLine("Creation: {0}", f.CreationTime);
Console.WriteLine("Attributes: {0}", f.Attributes.ToString());}
      }
```



```
 using System;
using System.Text;
using System.IO;       //для работы с файловым вводом-выводом
using System.Text.RegularExpressions;

namespace ConsoleApplication1    {
   class Program      {
      //  пример по удалению файлов:
      static void printFile(FileInfo file)         {
         Console.WriteLine("***** " + file.Name + " *****");
         Console.WriteLine("File size: {0}", file.Length);
         Console.WriteLine("Creation: {0}", file.CreationTime);
         Console.WriteLine("Attributes: {0}", file.Attributes.ToString());
      }
      static void Main(string[] args)         {
         DirectoryInfo dir = new DirectoryInfo(@"d:\БГУИР\ggg");
         FileInfo[] files = dir.GetFiles();
         if (files.Length != 0)
         {
            Console.WriteLine("Найдено {0} файла", files.Length);
            foreach (FileInfo f in files)
            {           printFile(f);                  f.Delete();              }
            Console.WriteLine("\nТеперь в каталоге содержится {0} файлов и можно его
    удалить",
            dir.GetFiles().Length);
            dir.Delete();
         }
      }     }   }
```


Компьютер ▸ Съемный диск (D:) ▸ БГУИР ▸ ggg
Упорядочить
Открыть
Печать
Новая папка
Избранное
Загрузки
Недавние места
Рабочий стол
Рабочий стол
Библиотеки
user
Компьютер
Дисковод (A:)
Локальный диск (
Съемный диск (D:
Сеть
Панель управления
Корзина
Info
AiP
C++
Имя
ЛАБОРАТОРНАЯ РАБОТА 7.d
лаб раб 1.docx
лаб раб 2.docx
ЛАБОРАТОРНАЯ РАБОТА 1.d
ЛАБОРАТОРНАЯ РАБОТА 6.d
ЛАБОРАТОРНАЯ РАБОТА 6.d
Выбрано элементов: 6
C:\Windows\system32\cmd.exe
Найдено 6 файла
***** ЛАБОРАТОРНАЯ РАБОТА 7.docx *****
File size: 14262
Creation: 06.02.2012 14:26:32
Attributes: Archive
***** лаб раб 1.docx *****
File size: 40047
Creation: 06.02.2012 14:26:33
Attributes: Archive
***** лаб раб 2.docx *****
File size: 30878
Creation: 06.02.2012 14:26:33
Attributes: Archive
***** ЛАБОРАТОРНАЯ РАБОТА 1.docx *****
File size: 53382
Creation: 06.02.2012 14:26:33
Attributes: Archive
***** ЛАБОРАТОРНАЯ РАБОТА 6.doc *****
File size: 21504
Creation: 06.02.2012 14:26:33
Attributes: Archive
***** ЛАБОРАТОРНАЯ РАБОТА 6.docx *****
File size: 11853
Creation: 06.02.2012 14:26:33
Attributes: Archive
Теперь в каталоге содержится 0 файлов и можно его удалить

Класс File

- Доступ к физическим файлам можно получать и через статические методы класса File.

```
static void Main(string[] args)
{
File.Copy(@"d:\prim\letter\letter1.txt",@"d:\prim\bmp\letter1.txt");
Directory.CreateDirectory(@"d:\prim\archives");
File.Move(@"d:\prim\letter\letter1.txt",@"d:\prim\archives\letter1.txt");
File.Delete(@"d:\prim\letter\letter2.txt");
Directory.Delete(@"d:\prim\letter");
}
```

- Имеет прямой смысл использовать статический класс File, когда требуется осуществить единственный вызов метода на объект (вызов будет выполнен быстрее, т. к..NET Framework не придется проходить через процедуру создания экземпляра нового объекта с последующим вызовом метода).

- Если приложение осуществляет несколько операций над файлом, то более разумным представляется создать экземпляр объекта FileInfo и использовать его методы (это позволит сэкономить определенное время, поскольку объект будет заранее настроен на нужный файл в файловой системе, в то время как статическому классу придется каждый раз осуществлять его поиск заново)

- Аналогичное правило действует и при выборе между классами Directory и DirectoryInfo.