

# **Работа с дисками и каталогами**

# Работа с дисками

Для представления диска в пространстве имен `System.IO` имеется класс `DriveInfo`.

Этот класс имеет метод `GetDrives()`, который возвращает имена всех логических дисков компьютера.

# Работа с дисками

Класс `DriveInfo` также имеет ряд полезных свойств:

- `AvailableFreeSpace`: кол-во доступного свободного места на диске в байтах
- `DriveFormat`: получает имя файловой системы
- `DriveType`: представляет тип диска
- `IsReady`: готов ли диск

# Работа с дисками

- **Name:** имя диска
- **RootDirectory:** возвращает корневой каталог диска
- **TotalFreeSpace:** получает общий объем свободного места на диске в байтах
- **TotalSize:** размер диска в байтах
- **VolumeLabel:** получает или устанавливает метку тома

# Пример

```
DriveInfo[] drives = DriveInfo.GetDrives();  
    foreach (DriveInfo drive in drives){  
        Console.WriteLine($"Название: {drive.Name}");  
        Console.WriteLine($"Тип: {drive.DriveType}");  
        if (drive.IsReady){  
            Console.WriteLine($"Объем диска: {drive.TotalSize}");  
            Console.WriteLine($"Свободное пространство: {drive.TotalFreeSpace}");  
            Console.WriteLine($"Метка диска: {drive.VolumeLabel}");  
        }  
        Console.WriteLine();  
    }
```

# Пример

```
DriveInfo drive = new DriveInfo("C");  
if (drive.IsReady){  
    Console.WriteLine($"Имя диска: {drive.Name}");  
    Console.WriteLine($"Тип диска: {drive.DriveType}");  
    Console.WriteLine($"Файловая система: {drive.DriveFormat}");  
    Console.WriteLine($"Общий объем: {drive.TotalSize}");  
    Console.WriteLine($"Свободное место: {drive.AvailableFreeSpace}");  
    Console.WriteLine($"Всего свободного места: {drive.TotalFreeSpace}");  
}
```

Имя диска: C:\

Тип диска: Fixed

Файловая система: NTFS

Общий объем: 480085274624

Свободное место: 38643552256

Всего свободного места: 38643552256

# Работа с каталогами

Термины «папка», «каталог» и «директория» в контексте файловой системы являются синонимами и взаимозаменяемыми. Все эти термины обозначают одно и то же — структуру для организации файлов и других каталогов внутри файловой системы.



# Работа с каталогами

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

- `CreateDirectory(path)`: создает каталог
- `Delete(path)`: удаляет каталог
- `Exists(path)`: определяет, существует ли каталог
- `GetCurrentDirectory()`: получает путь к текущей папке

# Работа с каталогами

- `GetDirectories(path)`: получает список подкаталогов
- `GetFiles(path)`: получает список файлов
- `GetFileSystemEntries(path)`: получает список подкаталогов и файлов в каталоге `path`
- `Move(sourceDirName, destDirName)`: перемещает каталог
- `GetParent(path)`: получение родительского каталога

# Работа с каталогами

- `GetLastWriteTime(path)`: возвращает время последнего изменения каталога
- `GetLastAccessTime(path)`: возвращает время последнего обращения к каталогу
- `GetCreationTime(path)`: возвращает время создания каталога

# Пример:

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

# Пример:

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

# Работа с каталогами

Класс `DirectoryInfo` предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на `Directory`, но не является статическим.

# Работа с каталогами

Класс `DirectoryInfo` предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. Во многом он похож на `Directory`, но не является статическим.

# Работа с каталогами

Для создания объекта класса `DirectoryInfo` применяется конструктор, который в качестве параметра принимает путь к каталогу:

```
public DirectoryInfo (string path);
```



# Работа с каталогами

Основные методы класса DirectoryInfo:

- `Create()`: создает каталог
- `CreateSubdirectory(path)`: создает подкаталог по указанному пути path
- `Delete()`: удаляет каталог
- `GetDirectories()`: получает список подкаталогов папки в виде массива DirectoryInfo
- `GetFiles()`: получает список файлов в папке в виде массива FileInfo
- `MoveTo(destDirName)`: перемещает каталог

# Работа с каталогами

Основные свойства класса DirectoryInfo:

- **CreationTime**: представляет время создания каталога
- **LastAccessTime**: представляет время последнего доступа к каталогу
- **LastWriteTime**: представляет время последнего изменения каталога
- **Exists**: определяет, существует ли каталог

# Работа с каталогами

- **Parent:** получение родительского каталога
- **Root:** получение корневого каталога
- **Name:** имя каталога
- **FullName:** полный путь к каталогу

# Что выбрать?

Используйте **Directory**, когда вам нужно выполнить одноразовые операции с директориями, такие как создание, удаление или получение списка файлов и подпапок.

Подходит для простых и быстрых операций, где сохранение состояния не требуется.

# Что выбрать?

Используйте **DirectoryInfo**, когда вам нужно работать с одной и той же директорией неоднократно, сохраняя её состояние.

Полезен в сложных сценариях, где требуется многократное обращение к одной и той же директории или работа с несколькими объектами одновременно.

# Фильтрация файлов и папок

// класс Directory

```
string[] dirs = Directory.GetDirectories(dirName, "books*.");
```

// класс DirectoryInfo

```
var directory = new DirectoryInfo(dirName);
```

```
DirectoryInfo[] dirs = directory.GetDirectories("books*.");
```

# Пример

```
string path = @"C:\Temp";  
string subpath = @"ISPP\LabWork1";  
DirectoryInfo dirInfo = new DirectoryInfo(path);  
if (!dirInfo.Exists){  
    dirInfo.Create();  
}  
dirInfo.CreateSubdirectory(subpath);
```

# Пример

```
string path = @"C:\SomeDir";  
string subpath = @"program\avalon";  
if (!Directory.Exists(path)){  
    Directory.CreateDirectory(path);  
}  
Directory.CreateDirectory($"{path}/{subpath}")
```



# Практическое задание

- Вывести информацию о всех дисках в системе
- Вывести диск с наибольшим свободным пространством
- Предоставить пользователю возможность создания директории по указанному пути
- Удалять указанный пользователем файл
- Вывести все папки и файлы указанной директории