Работа с дисками

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

Класс **DriveInfo** имеет статический метод **GetDrives**, который возвращает имена всех логических дисков компьютера. Также он предоставляет ряд полезных свойств:

- **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.

Получение имен и свойства всех дисков на компьютере:

```
DriveInfo[] drives = DriveInfo.GetDrives();

    foreach (DriveInfo drive in drives)
    {
        Console.WriteLine("Название: {0}", drive.Name);
        Console.WriteLine("Тип: {0}", drive.DriveType);
        if (drive.IsReady)
        {
            Console.WriteLine("Объем диска: {0}", drive.TotalSize);
            Console.WriteLine("Свободное пространство: {0}", drive.TotalFreeSpace);
            Console.WriteLine("Метка: {0}", drive.VolumeLabel);
        }
```

Работа с каталогами

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 DirectoryInfo предоставляет функциональность для создания, удаления, перемещения и других операций с каталогами. 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

Получение списка файлов и подкаталогов

```
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);     }
}
```

Создание каталога

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

Получение информации о каталоге

```
string dirName = "C:\\Program Files";

DirectoryInfo dirInfo = new DirectoryInfo(dirName);

Console.WriteLine("Название каталога: {0}", dirInfo.Name);
Console.WriteLine("Полное название каталога: {0}",
dirInfo.FullName);
Console.WriteLine("Время создания каталога: {0}",
dirInfo.CreationTime);
Console.WriteLine("Корневой каталог: {0}", dirInfo.Root);
```

Удаление каталога

```
string dirName = @"C:\SomeFolder";

try
{
    DirectoryInfo dirInfo = new DirectoryInfo(dirName);
    dirInfo.Delete(true);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

Перемещение каталога

```
string oldPath = @"C:\SomeFolder";
string newPath = @"C:\SomeDir";
DirectoryInfo dirInfo = new DirectoryInfo(oldPath);
if (dirInfo.Exists && Directory.Exists(newPath) == false)
{
    dirInfo.MoveTo(newPath);
}
```

Работа с файлами. Классы File и FileInfo

Некоторые полезные методы и свойства класса FileInfo:

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 Directory: получает родительский каталог в виде объекта DirectoryInfo
Свойство DirectoryName: получает полный путь к родительскому каталогу
Свойство Exists: указывает, существует ли файл
Свойство Length: получает размер файла
Свойство Extension: получает расширение файла
Свойство Name: получает имя файла
Свойство FullName: получает полное имя файла

Класс File реализует функциональность с помощью статических методов:

Copy(): копирует файл в новое место

Create(): создает файл

Delete(): удаляет файл

Move: перемещает файл в новое место

Exists(file): определяет, существует ли файл

Получение информации о файле

```
string path = @"C:\apache\hta.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);
}
```

Удаление файла

```
string path = @"C:\apache\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
   fileInf.Delete();
   // альтернатива с помощью класса File
   // File.Delete(path);
}
```

Перемещение файла

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
  fileInf.MoveTo(newPath);
  // альтернатива с помощью класса File
  // File.Move(path, newPath);
}
```

Копирование файла

```
string path = @"C:\apache\hta.txt";
string newPath = @"C:\SomeDir\hta.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
  fileInf.CopyTo(newPath, true);
  // альтернатива с помощью класса File
  // File.Copy(path, newPath, true);
}
```

Чтение из файла и StreamReader

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла. Среди его методов можно выделить следующие:

Close: закрывает считываемый файл и освобождает все ресурсы

Peek: возвращает следующий доступный символ, если символов больше нет, то возвращает -1

Read: считывает и возвращает следующий символ в численном представлении. Имеет перегруженную версию: Read(char[] array, int index, int count), где array - массив, куда считываются символы, index - индекс в массиве array, начиная с которого записываются считываемые символы, и count - максимальное количество считываемых символов

ReadLine: считывает одну строку в файле

ReadToEnd: считывает весь текст из файла

```
Console.WriteLine("******считываем весь файл********");
    using (StreamReader sr = new StreamReader(path))
    {
        Console.WriteLine(sr.ReadToEnd());
    }


Console.WriteLine("******считываем построчно********");
    using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
    {
        string line;
        while ((line = sr.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
    }
```

```
Console.WriteLine("******считываем блоками********");
   using (StreamReader sr = new StreamReader(path,
System.Text.Encoding.Default))
   {
      char[] array = new char[4];
      // считываем 4 символа
      sr.Read(array, 0, 4);

      Console.WriteLine(array);
   }
```

Запись в файл и StreamWriter

Для записи в текстовый файл используется класс StreamWriter. Свою функциональность он реализует через следующие методы:

Close: закрывает записываемый файл и освобождает все ресурсы

Flush: записывает в файл оставшиеся в буфере данные и очищает буфер.

Write: записывает в файл данные простейших типов, как int, double, char, string и т.д.

WriteLine: также записывает данные, только после записи добавляет в файл символ окончания строки

```
using (StreamReader sr = new StreamReader(readPath, System.Text.Encoding.Default))
    {
        text=sr.ReadToEnd();
    }
    using (StreamWriter sw = new StreamWriter(writePath, false,
System.Text.Encoding.Default))
    {
        sw.WriteLine(text);
    }

    using (StreamWriter sw = new StreamWriter(writePath, true,
System.Text.Encoding.Default))
    {
        sw.WriteLine("Дозапись");
        sw.Write(4.5);
    }
```

- 1. Дан текстовый файл, содержащий целые числа. Удалить из него все четные числа.
- 2. В данном текстовом файле удалить все слова, которые содержат хотя бы одну цифру.
- 3. Дан текстовый файл. Создать новый файл, каждая строка которого получается из соответствующей строки исходного файла перестановкой слов в обратном порядке.
- 4. Дан текстовый файл. Создать новый файл, состоящий из тех строк исходного файла, из чисел которых можно составить арифметическую прогрессию.
- 5. Даны два текстовых файла, содержащие целые числа. Создать файл из различных чисел, которые содержатся: а) в каждом исходном файле; б) только в одном из двух исходных файлов; в) только в первом исходном файле; г) хотя бы в одном из двух исходных файлов.
- 6. Создать и заполнить файл случайными целыми значениями. Выполнить сортировку содержимого файла по возрастанию.
- Создать типизированный файл записей со сведениями о телефонах абонентов; каждая запись имеет поля: фамилия абонента, год установки телефона, номер телефона. По заданной фамилии абонента выдать номера его телефонов. Определить количество установленных телефонов с N-го года.
- В текстовый файл занесены пары чисел, разделенных пробелом (каждая пара чисел – в новой строке). Рассматривая каждую пару как координаты точек на плоскости, найти наибольшее и наименьшее расстояния между этими точками.

- Имеется файл с текстом. Осуществить шифрование данного текста в новый файл путем записи текста в матрицу символов по строкам, а затем чтение символов из этой матрицы по столбцам. Осуществить расшифровку полученного текста.
- Создать программу, переписывающую в текстовый файл g содержимое файла f, исключая пустые строки, а остальные дополнить справа пробелами или ограничить до n символов.
- В файле, содержащем фамилии студентов и их оценки, изменить на прописные буквы фамилии тех студентов, которые имеют средний балл за национальной шкалой более «4».
- Из текстового файла удалить все слова, содержащие от трех до пяти символов, но при этом из каждой строки должно быть удалено только четное количество таких слов.
- Получить файл g, состоящий из строк файла f, содержащих заданную строку S. Предусмотреть случай, когда строка размещается в двух строках файла «с переносом».
- Получить файл g, в котором текст выровнен по правому краю путем равномерного добавления пробелов.
- Из текста программы выбрать все числа (целые и вещественные) и записать их в файл g в виде: число 1 – номер строки, число 2 – номер строки и так далее.
- Определить, симметричен ли заданный во входном файле текст.