Основы программирования

Средства ввода-вывода

Объекты файловой системы

...

- Большинство задач в программировании так или иначе связаны с работой с файлами и каталогами.
- Может потребоваться прочитать текст из файла или наоборот произвести запись, удалить файл или целый каталог, не говоря уже о разработке комплексных приложений, таких как например, создание текстового редактора или других подобных задач.

...

- В .NET существует возможность по управлять и манипулировать файлами и каталогами.
- Основные классы сосредоточены в пространстве имен System.IO.
- Классы, расположенные в этом пространстве имен (такие как Stream, StreamWriter, FileStream и др.), позволяют управлять файловым вводом-выводом.

Общие замечания

К объектам файловой системы относятся:

- диски,
- каталоги (папки, директории),
- файлы.

Объекту файловой системы ставится в соответствие объект определенного класса.

Через этот объект можно программно управлять объектом файловой системы.

Классы для работы с файловой системой

- `DriveInfo` для работы с дисками `(System.IO)`.
- `DirectoryInfo` и `Directory` для работы с каталогами `(System.IO)`.
- `FileInfo` и `File` для работы с файлами `(System.IO)`.

Классы `DirectoryInfo` и `FileInfo` являются наследниками абстрактно класса `FileSystemInfo`, каждый из них по-своему реализует унаследованные абстрактные методы.

Класс DriveInfo. Методы

- `GetDrives()` – статический метод, возвращает имена всех логических дисков компьютера. Возвращает массив с информацией о логических дисках ПК.
- `ToString()` – возвращает имя диска в виде строки.

Класс DriveInfo. Свойства

- `AvailableFreeSpace` – указывает объем доступного свободного места на диске в байтах;
- `DriveFormat` – возвращает имя файловой системы;
- `DriveType` – возвращает тип диска;
- `IsReady` – возвращает значение, указывающее, готов ли диск (например, DVD-диск может быть не вставлен в дисковод);
- `Name` – возвращает имя диска;
- `TotalFreeSpace` – возвращает общий объем свободного места на диске в байтах;

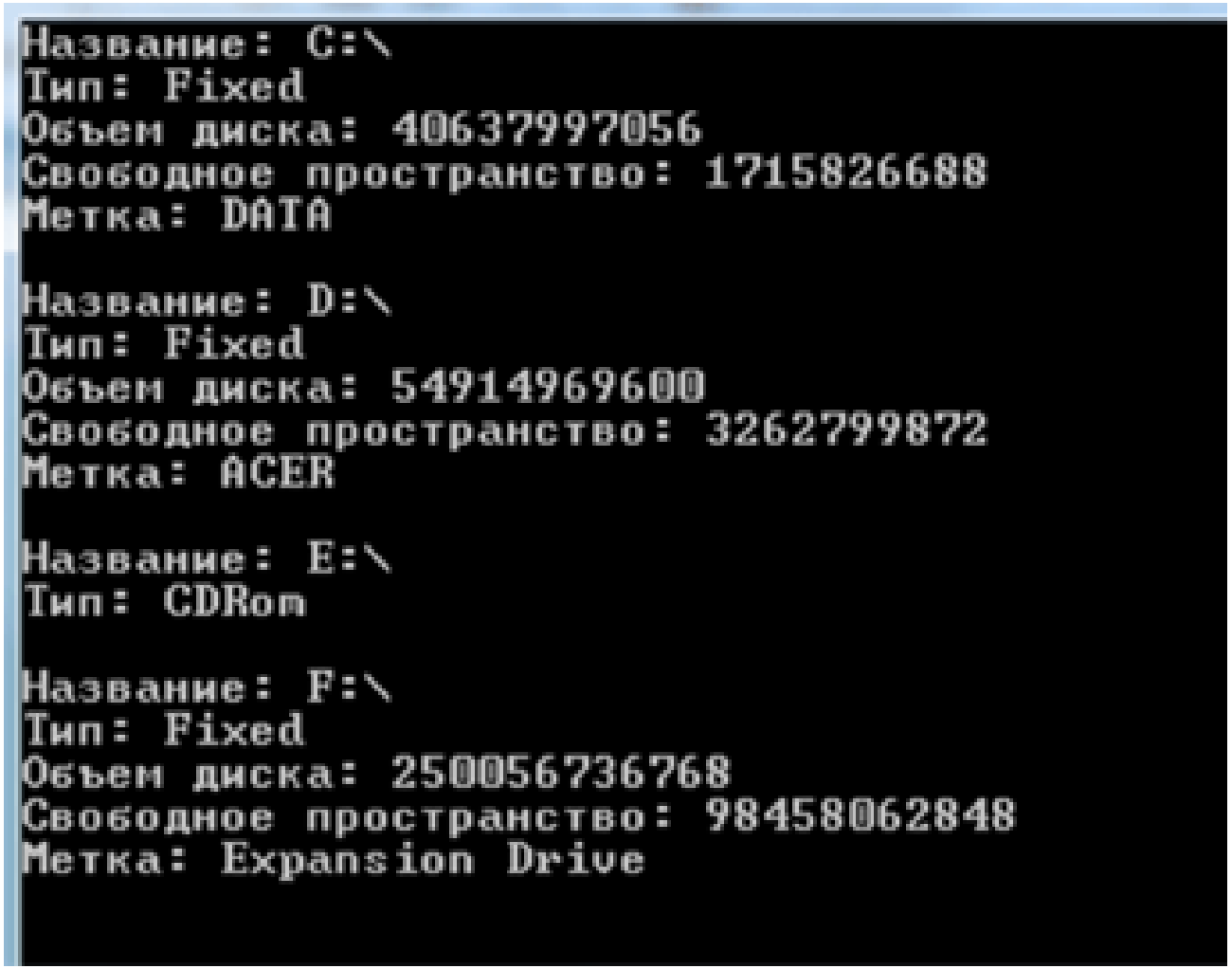
Класс DriveInfo. Свойства

- `TotalSize` – возвращает общий размер диска в байтах;
- `VolumeLabel` – возвращает или устанавливает метку тома.

Класс DriveInfo. Пример

```
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); }
    Console.WriteLine();                    }
Console.ReadLine();
```

Класс DriveInfo. Пример



Класс FileSystemInfo

Сначала рассмотрим класс `FileSystemInfo`, так как `DirectoryInfo` и `FileInfo` (будет рассмотрен позже) являются наследниками этого класса.

Класс FileSystemInfo. Поля

- `FullPath` – полный путь к каталогу или файлу;
- `OriginalPath` – первоначально заданный пользователем относительный или абсолютный путь.

Класс FileSystemInfo. Основные свойства

- `Attributes` – возвращает или устанавливает атрибуты текущего файла или каталога (задается из перечисления `FileAttributes`);
- `CreationTime` – возвращает или устанавливает временя создания текущего объекта файловой системы;
- `Exists` – возвращает значение, показывающее, существует ли данный файл или каталог;
- `Extension` – возвращает строку, содержащую расширение файла;
- `FullName` – возвращает полный путь к текущему объекту файловой системы;

Класс FileSystemInfo. Основные свойства

- `LastAccessTime` – возвращает или устанавливает время последнего время последнего доступа к текущему файлу или каталогу;
- `LastWriteTime` – возвращает или устанавливает время последней операции записи в текущий файл или каталог;
- `Name` – возвращает имя файла или имя последнего каталога в текущей иерархии (текущего каталога);
- …

Класс FileSystemInfo. Некоторые методы

- `Delete()` – удаляет файл или каталог;
- `Refresh()` – обновляет состояние объекта;
- `ToString` – возвращает исходный путь.
  Для получения полного пути или имени файла или каталога рекомендуется использовать свойства `FullName` или `Name`.
- …

Классы DirectoryInfo и Directory

Для работы с каталогами в пространстве имен `System.IO` предназначены сразу два класса:

- `Directory` – предоставляет статические методы для создания, перемещения и перечисления в каталогах и вложенных каталогах;
- `DirectoryInfo` – предоставляет методы экземпляра класса для создания, перемещения и перечисления в каталогах и подкаталогах. Является наследником класса `FileSystemInfo`.

Класс DirectoryInfo. Свойства

- `Exists` – возвращает значение, определяющее наличие каталога `true` если каталог существует и `false` если нет;
- `Name` – возвращает имя данного каталога;
- `Parent` – возвращает ссылку на родительский каталог;
- `Root` – возвращает корневую часть пути к каталогу.

Класс DirectoryInfo. Некоторые методы

- `Create()` – создает каталог по указанному пути.
- `CreateSubdirectory(path)` – создает один или несколько подкаталогов по заданному в параметре пути.
- `Delete()` – удаляет каталог. Вариант `Delete(bool)`.
- `GetDirectories()` – возвращает все вложенные подкаталоги текущего каталога в виде массива строковых имен.
- `GetFiles()` – возвращает все вложенные файлы в виде массива объектов.
- `MoveTo(destDirName)` – перемещает каталог и его содержимого в новое местоположение.

Класс Directory. Некоторые методы

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

Пример. Создание каталога

```
1  string path = @"C:\SomeDir";
2  string subpath = @"program\avalon";
3  DirectoryInfo dirInfo = new DirectoryInfo(path);
4  if (!dirInfo.Exists)
5  {
6      dirInfo.Create();
7  }
8  dirInfo.CreateSubdirectory(subpath);
```

Проверяем есть ли такая директория.
Если она существует, то ее создать будет нельзя и приложение выбросит ошибку.

В итоге должен получиться путь: `"C:\SomeDir\program\avalon"`

Пример. Перемещение каталога

```
1  string oldPath = @"C:\SomeFolder";
2  string newPath = @"C:\SomeDir";
3  DirectoryInfo dirInfo = new DirectoryInfo(oldPath);
4  if (dirInfo.Exists && Directory.Exists(newPath) == false)
5  {
6      dirInfo.MoveTo(newPath);
7  }
```

При перемещении следует учитывать, что новый каталог, в который мы хотим перемесить все содержимое старого каталога, не должен существовать.

Работа с файлами

- Для работы с файлами предназначена пара классов `File` и `FileInfo`.
- С их помощью мы можем создавать, удалять, перемещать файлы, получать их свойства и многое другое.

Классы FileInfo и File

Для работы с файлами в пространстве имен `System.IO` предназначены сразу два класса:

- `File` – предоставляет статические методы создания, перемещения, удаления файлов, получения их свойств и другие;
- `FileInfo` – предоставляет методы экземпляра класса для создания, перемещения, удаления и открытия файлов. Является наследником класса `FileSystemInfo`.

Класс FileInfo. Некоторые свойства

- `Directory` – возвращает ссылку на каталог, в котором находится файл.
- `DirectoryName` – возвращает полный путь к каталогу, в котором находится файл.
- `Exists` – возвращает значение, показывающее существует ли файла: значение `true` если файл существует и `false` если нет.
- `Length` – возвращает размер файла в байтах.
- `Name` – возвращает имя файла.

Класс FileInfo. Методы

- `AppendText()` – открывает файл для дописывания в текстовом режиме, возвращает поток `StreamWriter`.
- `CopyTo(path)` – копирует существующий файл в новый по указанному пути `path`.
- `Create()` – создает новый файл и возвращает поток `FileStream`.
- `CreateText()` – создает новый текстовый файл и возвращает поток `StreamWriter`.
- `Delete()` – удаляет файл.
- `MoveTo(destFileName)` – перемещает файл в новое место, разрешая переименование файла.

Класс FileInfo. Методы

- `Open(FileMode)` – открытие файла с различными правами на чтение и запись и привилегиями для совместной работы. Возвращает поток `StreamWriter`.
- `OpenRead()` – открытие файла только для чтения в байтовом режиме. Возвращает поток `FileStream`.
- `OpenText()` – открывает существующий текстовый файл для чтения. Возвращает поток `StreamReader`.
- `OpenWrite()` – открытие файла для записи в байтовом режиме. Возвращает поток `FileStream`.

Класс File

Класс `File` реализует похожую функциональность с помощью статических методов (некоторых):

- `Copy(sourceFileName, destFileName)` – копирует файл в новое место.
- `Create(path)` – создает файл. Параметр: путь и имя создаваемого файла. Возвращает поток `FileStream`.
- `Delete(path)` – удаляет файл с заданным именем.
- `Exists(path)` – определяет, существует ли файл.
- `Move(sourceFileName, destFileName)` – перемещает файл в новое место.
- …

Пример. Информации о файле

```
1  string path = @"C:\apache\hta.txt";
2  FileInfo fileInf = new FileInfo(path);
3  if (fileInf.Exists)
4  {
5      Console.WriteLine("Имя файла: {0}", fileInf.Name);
6      Console.WriteLine("Время создания: {0}", fileInf.CreationTime);
7      Console.WriteLine("Размер: {0}", fileInf.Length);
8  }
```

Пример. Удаление файла

```
1  string path = @"C:\apache\hta.txt";
2  FileInfo fileInf = new FileInfo(path);
3  if (fileInf.Exists)
4  {
5      fileInf.Delete();
6      // альтернатива с помощью класса File
7      // File.Delete(path);
8  }
```

Пример. Перемещение файла

```
1  string path = @"C:\apache\hta.txt";
2  string newPath = @"C:\SomeDir\hta.txt";
3  FileInfo fileInf = new FileInfo(path);
4  if (fileInf.Exists)
5  {
6      fileInf.MoveTo(newPath);
7      // альтернатива с помощью класса File
8      // File.Move(path, newPath);
9  }
```

Пример. Копирование файла

```
1  string path = @"C:\apache\hta.txt";
2  string newPath = @"C:\SomeDir\hta.txt";
3  FileInfo fileInf = new FileInfo(path);
4  if (fileInf.Exists)
5  {
6      fileInf.CopyTo(newPath, true);
7      // альтернатива с помощью класса File
8      // File.Copy(path, newPath, true);
9  }
```

Пример. Копирование файла

- Метод `CopyTo` класса `FileInfo` принимает два параметра: путь, по которому файл будет копироваться, и булевое значение, которое указывает, надо ли при копировании перезаписывать файл (если `true`, как в случае выше, файл при копировании перезаписывается). Если же в качестве последнего параметра передать значение `false`, то если такой файл уже существует, приложение выдаст ошибку.
- Метод `Copy` класса `File` принимает три параметра: путь к исходному файлу, путь, по которому файл будет копироваться, и булевое значение, указывающее, будет ли файл перезаписываться.

Перечисление FileAttributes

Описывает атрибуты объекта файловой системы.

- `Archive` – файл используется при выполнении резервного копирования.
- `Compressed` – сжатый файл.
- `Directory` – объект файловой системы является каталогом.
- `Encrypted` – шифрованный файл.
- `Hidden` – скрытый файл.

Перечисление FileAttributes

- `Normal` – обычный файл. Не используется совместно с другими атрибутами.
- `Offline` – серверный файл, кэширован в хранилище на клиентском компьютере.
- `ReadOnly` – файл доступен только для чтения
- `System` – системный файл.
- …

Перечисление FileMode

Описывает режимы открытия файла.

Одно из значений перечисления передается в качестве параметра в методы открытия файла.

Перечисление предназначено для гибкой настройки режима работы с файлом.

Перечисление FileMode

- `Append` – открытие существующего файла с установкой указателя в конец файла. Если файл не существует, то создается новый файл.
- `Create` – создание нового файла, если файл уже существует, то его содержимое удаляется.
- `CreateNew` – создание нового файла, если файл уже существует, то будет выброшено исключение `IOException`.

Перечисление FileMode

- `Open` – открытие существующего файла, если файла не существует, то будет выброшено исключение `IOException`.
- `OpenOrCreate` – открытие существующего файла, если файла не существует, то он будет создан.
- `Truncate` – открытие существующего файла и удаление его содержимого.

Перечисление FileAccess

Описывает режимы доступа к файлу.

Одно из значений перечисления передается в качестве параметра в методы открытия файла.

Перечисление предназначено для гибкой настройки режима работы с файлом.

- `Read` – открытие файла только для чтения
- `ReadWrite` – открытие файла для чтения и записи
- `Write` – открытие файла только для записи

Перечисление FileShare

Описывает режимы совместного доступа к файлу.

Одно из значений перечисления передается в качестве параметра в методы открытия файла.

Перечисление предназначено для гибкой настройки режима работы с файлом.

Перечисление FileShare

- `Delete` – возможно последующее удаление файла.
- `None` – совместный доступ к файлу запрещен. Попытка открытия файла другим потоком приводит к ошибке.
- `Read` – совместный доступ к файлу нескольких потоков только для чтения.
- `ReadWrite` – совместный доступ к файлу нескольких потоков для чтения и записи.
- `Write` – совместный доступ к файлу нескольких потоков только для записи.

Перечисления FileMode, FileAccess, FileShare

```
FileStream s2 = new FileStream(name,
                    FileMode.Open,
                    FileAccess.Read,
                    FileShare.Read);
```

Источники

1. System.IO Пространство имен | Microsoft Docs https://docs.microsoft.com/ru-ru/dotnet/api/system.io?view=netcore-3.1
2. Работа с дисками https://metanit.com/sharp/tutorial/5.1.php
3. Работа с каталогами https://metanit.com/sharp/tutorial/5.2.php
4. Работа с файлами. Классы File и FileInfo https://metanit.com/sharp/tutorial/5.3.php