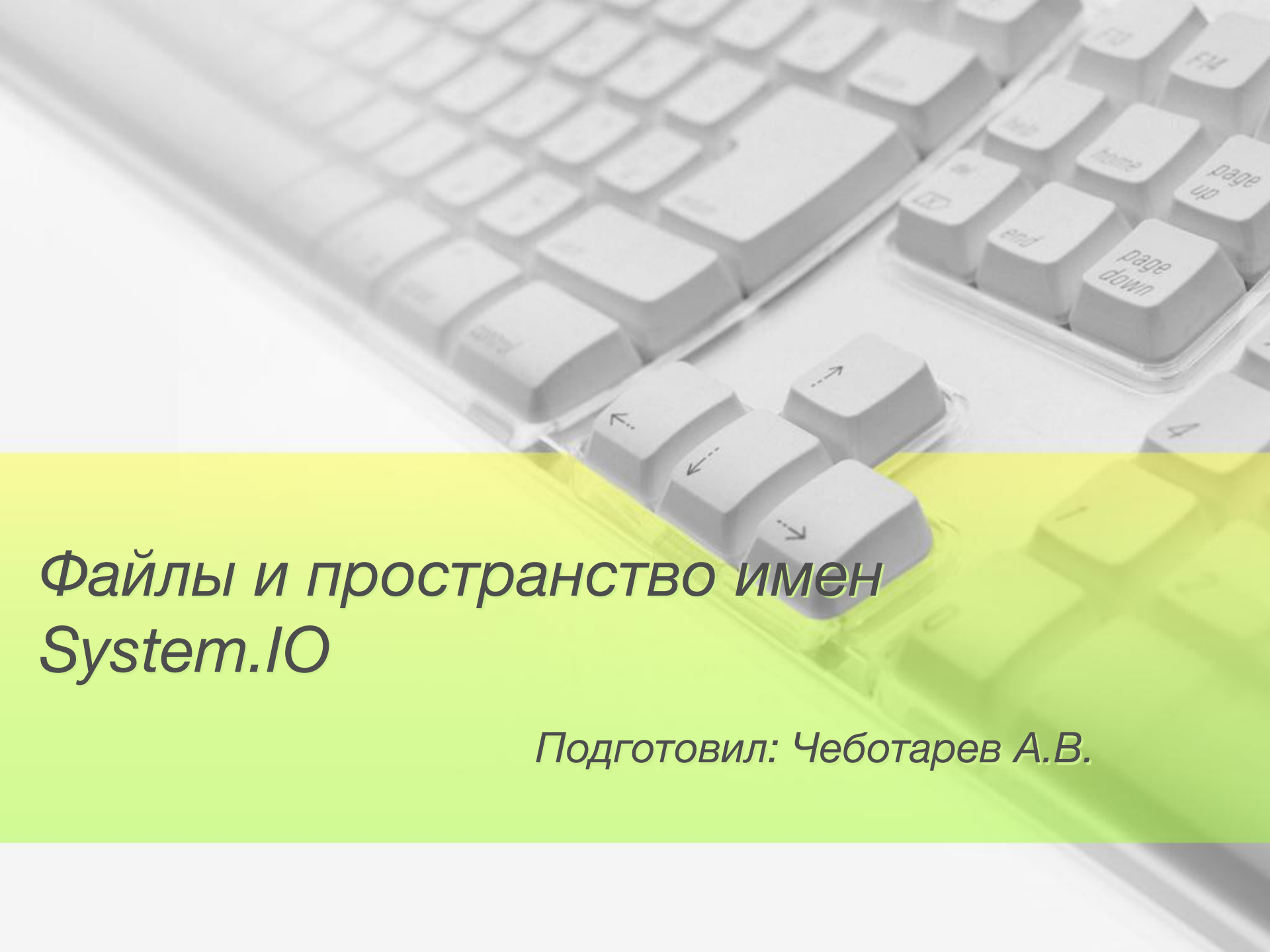

Файлы и пространство имен System.IO

Подготовил: Чеботарев А.В.

Основные понятия

- **Файл** — это набор данных, который хранится на внешнем запоминающем устройстве;
- **Поток** — это абстрактное представление данных (в байтах), которое облегчает работу с ними;
- **Папка** — экранный объект в графических интерфейсах операционных систем и программ, дающий доступ к каталогу файловой системы.

Пространство имен System.IO

- Пространство имен System.IO содержит типы, позволяющие осуществлять чтение и запись в файлы и потоки данных, а также типы для базовой поддержки файлов и папок.

Диск

Для работы с диском в System.IO имеется класс **DriveInfo**.

- **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

Диск

```
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);
    }
```

Каталоги

В пространстве имен System.IO классы: **Directory** и **DirectoryInfo**

Класс Directory

- **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

- **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

```
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);
```

Файлы

- Подобно паре Directory/DirectoryInfo для работы с файлами предназначена пара классов **File** и **FileInfo**.

Класс **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

- **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

```
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  }
```

Изменение файла

- Класс **FileStream** представляет возможности по считыванию из файла и записи в файл.

Изменение файла

- Свойство **Length**: возвращает длину потока в байтах
- Свойство **Position**: возвращает текущую позицию в потоке
- Метод **Read**: считывает данные из файла в массив байтов. Принимает три параметра: int Read(byte[] array, int offset, int count) и возвращает количество успешно считанных байтов.
- Метод **long Seek(long offset, SeekOrigin origin)**: устанавливает позицию в потоке со смещением на количество байт, указанных в параметре offset.
- Метод **Write**: записывает в файл данные из массива байтов. Принимает три параметра: Write(byte[] array, int offset, int count)

FileMode

- *Append* – открывает файл (если существует) и переводит указатель в конец файла (данные будут дописываться в конец), или создает новый файл. Данный режим возможен только при режиме доступа FileAccess.Write.
- *Create* - создает новый файл(если существует – заменяет)
- *CreateNew* – создает новый файл (если существует – генерируется исключение)
- *Open* - открывает файл (если не существует – генерируется исключение)
- *OpenOrCreate* – открывает файл, либо создает новый, если его не существует
- *Truncate* – открывает файл, но все данные внутри файла затирает (если файла не существует – генерируется исключение)

Чтение и запись текстовых файлов. StreamReader и StreamWriter

- Класс FileStream не очень удобно применять для работы с текстовыми файлами. К тому же для этого в пространстве System.IO определены специальные классы: **StreamReader** и **StreamWriter**.

StreamReader

- **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

StreamWriter

- **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

BinaryWriter и **BinaryReader**

- Для работы с бинарными файлами предназначена пара классов **BinaryWriter** и **BinaryReader**. Эти классы позволяют читать и записывать данные в двоичном формате.

BinaryWriter

- **Close()**: закрывает поток и освобождает ресурсы
- **Flush()**: очищает буфер, дописывая из него оставшиеся данные в файл
- **Seek()**: устанавливает позицию в потоке
- **Write()**: записывает данные в поток

BinaryReader

- **Close()**: закрывает поток и освобождает ресурсы
- **ReadBoolean()**: считывает значение bool и перемещает указать на один байт
- **ReadByte()**: считывает один байт и перемещает указать на один байт
- **ReadChar()**: считывает значение char, то есть один символ, и перемещает указать на столько байтов, сколько занимает символ в текущей кодировке
- **ReadDecimal()**: считывает значение decimal и перемещает указать на 16 байт
- **ReadDouble()**: считывает значение double и перемещает указать на 8 байт
- **ReadInt16()**: считывает значение short и перемещает указать на 2 байта
- **ReadInt32()**: считывает значение int и перемещает указать на 4 байта
- **ReadInt64()**: считывает значение long и перемещает указать на 8 байт
- **ReadSingle()**: считывает значение float и перемещает указать на 4 байта
- **ReadString()**: считывает значение string. Каждая строка предваряется значением длины строки, которое представляет 7-битное целое число

```
struct State
{
    public string name;
    public string capital;
    public int area;
    public double people;

    public State(string n, string c, int a, double p)
    {
        name = n;
        capital = c;
        people = p;
        area = a;
    }
}
```

```
State[] states = new State[2];
states[0] = new State("Германия", "Берлин",  357168,  80.8);
states[1] = new State("Франция", "Париж", 640679, 64.7);

string path= @"C:\SomeDir\states.dat";

try
{
    // создаем объект BinaryWriter
    using (BinaryWriter writer = new BinaryWriter(File.Open(path, FileMode.OpenOrCreate)))
    {
        // записываем в файл значение каждого поля структуры
        foreach (State s in states)
        {
            writer.Write(s.name);
            writer.Write(s.capital);
            writer.Write(s.area);
            writer.Write(s.people);
        }
```

```
    // создаем объект BinaryReader
    using (BinaryReader reader = new BinaryReader(File.Open(path, FileMode.Open)))
    {
        // пока не достигнут конец файла
        // считываем каждое значение из файла
        while (reader.PeekChar() > -1)
        {
            string name = reader.ReadString();
            string capital = reader.ReadString();
            int area = reader.ReadInt32();
            double population = reader.ReadDouble();

            Console.WriteLine("Страна: {0}  столица: {1}  площадь {2} кв. км   численность населения: {3} млн. чел.",
                name, capital, area, population);
        }
    }
}
catch (Exception e)
```