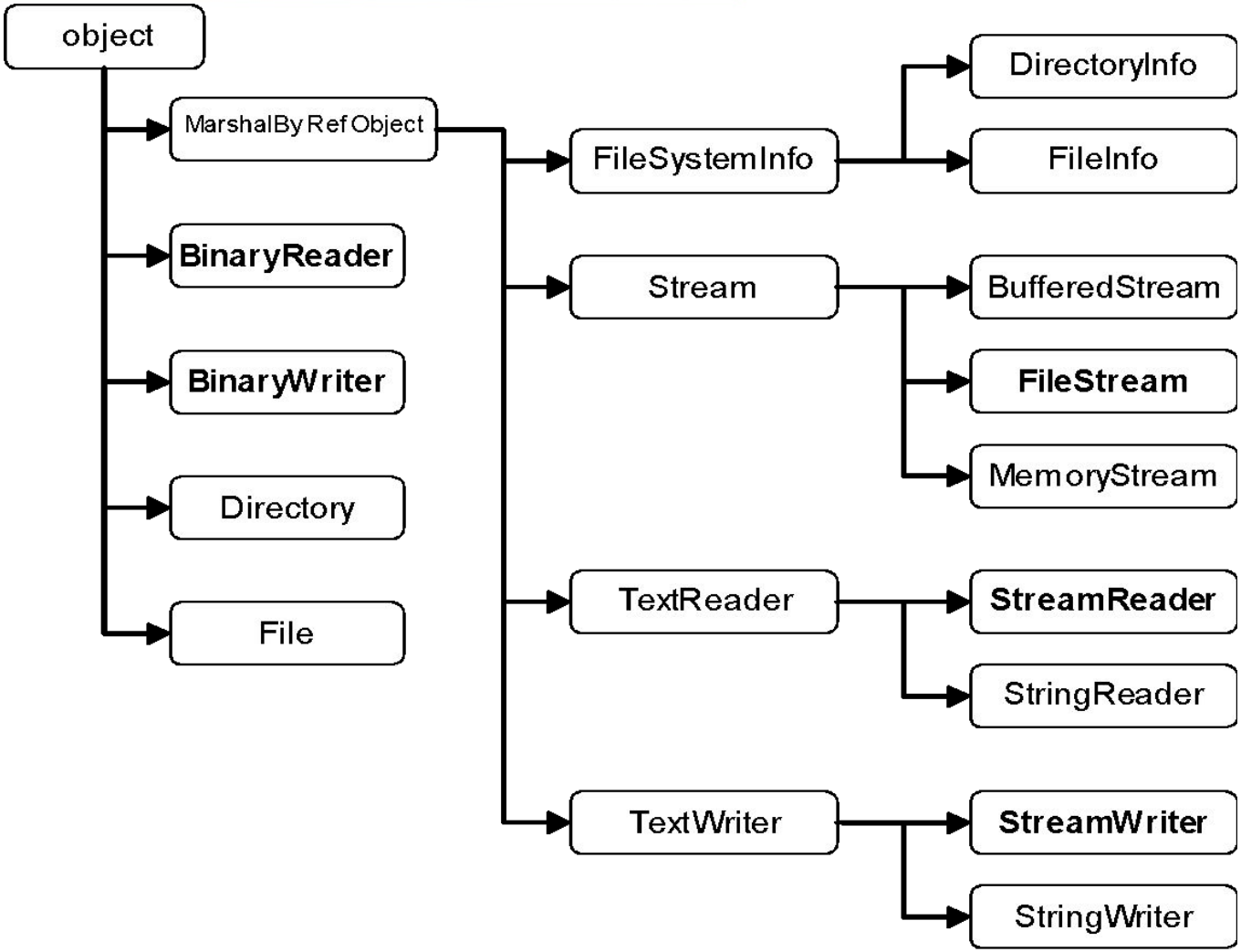
Работа с файлами

- ***Чтение** (ввод)* — передача данных с внешнего устройства в оперативную память, обратный процесс — ***запись** (вывод)*.
- Ввод-вывод в C# выполняется с помощью подсистемы ввода-вывода и классов библиотеки .NET. Обмен данными реализуется с помощью потоков.
- ***Поток*** (stream) — абстрактное понятие, относящееся к любому переносу данных от источника к приемнику. Потоки обеспечивают надежную работу как со стандартными, так и с определенными пользователем типами данных, а также единообразный и понятный синтаксис.
- Поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен.
- Обмен с потоком для повышения скорости передачи данных производится, как правило, через ***буфер***. Буфер выделяется для каждого открытого файла.

Классы .NET для работы с потоками



Выполнять обмен с внешними устройствами можно на уровне:

- *двоичного представления данных*

  (BinaryReader, BinaryWriter);

- *байтов*

  (FileStream);

- *текста*, то есть символов

  (StreamWriter, StreamReader).

Доступ к файлам

Доступ к файлам может быть:

последовательным - очередной элемент можно прочитать (записать) только после аналогичной операции с предыдущим элементом

произвольным, или ***прямым***, при котором выполняется чтение (запись) произвольного элемента по заданному адресу.

Текстовые файлы - только последовательный доступ

В двоичных и байтовых потоках можно использовать оба метода доступа

Прямой доступ в сочетании с отсутствием преобразований обеспечивает высокую скорость получения нужной информации.

Режимы доступа

Режимы доступа к файлу содержатся в перечислении **FileAccess**, **определенном в пространстве имен System.IO.**

Константы перечисления	
Значение	**Описание**
Read	**Открыть файл только для чтения**
ReadWrite	**Открыть файл для чтения и записи**
Write	**Открыть файл только для записи**

Режимы открытия

Значения перечисления FileMode

Значение	Описание
Append	Открыть файл, если он существует, и установить текущий указатель в конец файла. Если файл не существует, создать новый файл
Create	Создать новый файл. Если в каталоге уже существует файл с таким же именем, он будет стерт
CreateNew	Создать новый файл. Если в каталоге уже существует файл с таким же именем, возникает исключение IOException
Open	Открыть существующий файл
OpenOrCreate	Открыть файл, если он существует. Если нет, создать файл с таким именем
Truncate	Открыть существующий файл. После открытия он должен быть обрезан до нулевой длины

***Режимы совместного использования файла различными пользователями определяет* перечисление FileShare**

Значение	Описание
None	Совместное использование открытого файла запрещено. Запрос на открытие данного файла завершается сообщением об ошибке
Read	Позволяет открывать файл для чтения одновременно нескольким пользователям. Если этот флаг не установлен, запросы на открытие файла для чтения завершаются сообщением об ошибке
ReadWrite	Позволяет открывать файл для чтения и записи одновременно нескольким пользователям
Write	Позволяет открывать файл для записи одновременно нескольким пользователям

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

BinaryReader – класс Считывает простые типы данных как двоичные значения в заданной кодировке.

Элемент	Описание
Close	Закрывает текущий поток чтения и связанный с ним базовый поток.
Read(Byte[], Int32, Int32)	Считывает указанное количество байтов из потока, начиная с заданной точки в массиве байтов.
ReadByte	Считывает из текущего потока следующий байт и перемещает текущую позицию в потоке на один байт вперед.
ReadChar	Считывает следующий знак из текущего потока
ReadInt32	Считывает целое число со знаком длиной 4 байта
ReadDouble	Считывает число с плавающей запятой длиной 8 байт
ReadString	Считывает строку из текущего потока.

```
 double[] numbers = new double[20];
 using (var writer = new BinaryWriter(File.OpenWrite("file.bin")))
 {
     foreach (double number in numbers)
     writer.Write(number);
}
using (var reader = new BinaryReader(File.OpenRead("file.bin")))
{
   for (int i = 0; i < numbers.Length; i++)
   numbers[i] = reader.ReadDouble();
}
```

Потоки байтов

Ввод-вывод в файл на уровне байтов выполняется с помощью класса **FileStream**, который является наследником абстрактного класса Stream, определяющего набор стандартных операций с потоками.

Элемент	Описание
BeginRead, BeginWrite	Начать асинхронный ввод или вывод
Length	Возвратить длину потока в байтах
Read, ReadByte	Считать последовательность байтов (или один байт) из текущего потока и переместить указатель в потоке на количество считанных байтов
Seek	Установить текущий указатель потока на заданную позицию
Write, WriteByte	Записать последовательность байтов (или один байт) в текущий поток и переместить указатель в потоке на количество записанных байтов
Close	Закрыть текущий поток и освободить связанные с ним ресурсы (сокеты, указатели на файлы и т. п.)

```
FileStream f = new FileStream(@"TextFile1.txt", FileMode.Create,
FileAccess.ReadWrite);
        f.WriteByte(100); // начало файла записывается число 100

        byte[] x = new byte[10];
        for ( byte i = 0; i < 10; ++i )
        {
            x[i] = (byte)( 10 - i );
            f.WriteByte(i); // записывается 10 чисел от 0 до 9
        }

        f.Write( x, 0, 5 ); // записывается 5 элементов массива
        byte[] y = new byte[20];
        f.Seek( 0, SeekOrigin.Begin ); // текущий указатель - на начало
        f.Read( y, 0, 20 ); // чтение из файла в массив
        foreach ( byte elem in y ) Console.Write( " " + elem );
        Console.WriteLine();
        Console.ReadLine();
```

Символьные потоки StreamWriter и StreamReader работают с Unicode-символами

Элемент	Описание
Close	Закрыть файл и освободить связанные с ним ресурсы.
NewLine	Используется для задания последовательности символов, означающих начало новой строки.
Write	Записать фрагмент текста в поток
WriteLine	Записать строку в поток и перейти на другую строку
Read	Считать данные из входного потока
ReadBlock	Считать из входного потока указанное пользователем количество символов и записать их в буфер, начиная с заданной позиции
ReadLine	Считать строку из текущего потока и возвратить ее как значение типа string. Пустая строка (null) означает конец файла (EOF)
ReadToEnd	Считать все символы до конца потока, начиная с текущей позиции, и возвратить считанные данные как одну строку типа string

Пример чтения из текстового файла

```
static void Main()   // весь файл -> в одну строку
{      try
       {
          StreamReader f = new StreamReader( "text.txt" );
          string s = f.ReadToEnd();
          Console.WriteLine(s);
          f.Close();
       }
       catch( FileNotFoundException e )
       {
          Console.WriteLine( e.Message );
          Console.WriteLine( " Проверьте правильность имени файла!" );
          return;
       }
       catch
       {
                Console.WriteLine( " Неопознанное исключение!" );
           return;
       }
}
```

Построчное чтение текстового файла

```
StreamReader f = new StreamReader( "text.txt" );
string s;
long i = 0;
while ( ( s = f.ReadLine() ) != null ) Console.WriteLine( "{0}: {1}", ++i, s );
f.Close();
```

Чтение целых чисел из текстового файла

```
try {
      List<int> list_int = new List<int>();
          StreamReader file_in = new StreamReader( @"D:\FILES\1024" );
      Regex regex = new Regex( "[^0-9-+]+" );
      List<string> list_string = new List<string>(
                  regex.Split( file_in.ReadToEnd().TrimStart(' ') ) );
      foreach (string temp in list_string)
              list_int.Add( Convert.ToInt32(temp) );

      foreach (int temp in list_int) Console.WriteLine(temp);
      ...
}
catch (FileNotFoundException e)
      {  Console.WriteLine("Нет файла" + e.Message); return; }
catch (FormatException e)
      {  Console.WriteLine(e.Message); return;     }
catch { … }
```

Работа с каталогами и файлами

Пространство имен System.IO: классы

- Directory, DirectoryInfo
- File, FileInfo

(создание, удаление, перемещение файлов и каталогов, получение их свойств)

Элементы класса DirectoryInfo

Элемент	Описание
Create CreateSubDirectory	Создать каталог или подкаталог по указанному пути в файловой системе
Delete	Удалить каталог со всем его содержимым
GetDirectories	Возвратить массив строк, представляющих все подкаталоги
GetFiles	Получить файлы в текущем каталоге в виде массива объектов класса FileInfo
MoveTo	Переместить каталог и все его содержимое на новый адрес в файловой системе
Parent	Возвратить родительский каталог

Пример: копирование файлов *.jpg

```
class Class1                    // из каталога d:\foto в каталог d:\temp
  static void Main() {
    try {
        string destName = @"d:\temp\";
         DirectoryInfo dir = new DirectoryInfo( @"d:\foto" );
      if ( ! dir.Exists )        // проверка существования каталога
      {
          Console.WriteLine( "Каталог " + dir.Name + " не сущ." );
          return;
      }
         DirectoryInfo dest = new DirectoryInfo( destName );
         dest.Create();                  // создание целевого каталога
```

```
FileInfo[] files = dir.GetFiles( "*.jpg" ); //список файлов
foreach( FileInfo f in files )
f.CopyTo( dest + f.Name );        // копирование файлов
Console.WriteLine( "Скопировано файлов " + files.Length);
}        // конец блока try

 catch ( Exception e )
 {
   Console.WriteLine( "Error: " + e.Message );
 }
}
}
```