Теория и практика программирования

Шпилёв Пётр Валерьевич

Санкт-Петербургский государственный университет
Математико-механический факультет
Кафедра статистического моделирования

Лекция 7

Санкт-Петербург
2019 г.

e-mail: pitshp@hotmail.com

Таблица 1: Некоторые методы, определенные в классе Stream

Метод	Описание
void Close()	Закрывает поток
void Flush()	Выводит содержимое потока на физическое устройство
int ReadByte()	Возвращает целочисленное представление следующего байта, доступного для ввода из потока. При обнаружении конца файла возвращает значение -1
int Read(byte[] buffer, int offset, int count)	Делает попытку ввести count байтов в массив *buffer*, начиная с элемента *buffer[offset]*. Возвращает количество успешно введенных байтов
long Seek(long offset, SeekOrigin origin)	Устанавливает текущее положение в потоке по указанному смещению *offset* относительно заданного начала отсчета *origin*. Возвращает новое положение в потоке
void WriteByte(byte value)	Выводит один байт в поток вывода
void Write(byte[]buffer, int offset, int count)	Выводит подмножество count байтов из массива *buffer*, начиная с элемента *buffer[offset]*. Возвращает количество выведенных байтов

Таблица 2: Свойства, определенные в классе Stream

Свойство	Описание
bool CanRead	Принимает значение *true*, если из потока можно ввести данные. Доступно только для чтения
bool CanSeek	Принимает значение *true*, если поток поддерживает запрос текущего положения в потоке. Доступно только для чтения
bool CanWrite	Принимает значение *true*, если в поток можно вывести данные. Доступно только для чтения
long Length	Содержит длину потока. Доступно только для чтения
long Position	Представляет текущее положение в потоке. Доступно как для чтения, так и для записи
int ReadTimeout	Представляет продолжительность времени ожидания в операциях ввода. Доступно как для чтения, так и для записи
int WriteTimeout	Представляет продолжительность времени ожидания в операциях вывода. Доступно как для чтения, так и для записи

Таблица 3: Классы байтовых потоков

Класс потока	**Описание**
BufferedStream	Заключает в оболочку байтовый поток и добавляет буферизацию. Буферизация, как правило, повышает производительность
FileStream	Байтовый поток, предназначенный для файлового ввода-вывода
MemoryStream	Байтовый поток, использующий память для хранения данных
UnmanagedMemoryStream	Байтовый поток, использующий неуправляемую память для хранения данных

Таблица 4: Методы ввода, определенные в классе TextReader

Метод	Описание
int Peek()	Получает следующий символ из потока ввода, но не удаляет его. Возвращает значение -1, если ни один из символов не доступен
int Read()	Возвращает целочисленное представление следующего доступного символа из вызывающего потока ввода. При обнаружении конца потока возвращает значение -1
int Read(char[]buffer, int index, int count)	Делает попытку ввести количество count символов в массив buffer, начиная с элемента buffer[index], и возвращает количество успешно введенных символов
int ReadBlock(char[]buffer, int index, int count)	Делает попытку ввести количество count символов в массив buffer, начиная с элемента buffer[index], и возвращает количество успешно введенных символов
string ReadLine()	Вводит следующую текстовую строку и возвращает ее в виде объекта типа string. При попытке прочитать признак конца файла возвращает пустое значение
string ReadToEnd()	Вводит все символы, оставшиеся в потоке, и возвращает их в виде объекта типа string

Таблица 5: Методы класса TextWriter

Метод	Описание
void Write(int value)	Выводит значение типа int
void Write(double value)	Выводит значение типа double
void Write(bool value)	Выводит значение типа bool
void WriteLine(string value)	Выводит значение типа string с последующим символом новой строки
void WriteLine(uint value)	Выводит значение типа uint с последующим символом новой строки
void WriteLine(char value)	Выводит символ с последующим символом новой строки

Таблица 6: Классы символьных потоков, связанные с TextReader и TextWriter

Класс потока	Описание
StreamReader	Предназначен для ввода символов из байтового потока. Этот класс является оболочкой для байтового потока ввода
StreamWriter	Предназначен для вывода символов в байтовый поток. Этот класс является оболочкой для байтового потока вывода
StringReader	Предназначен для ввода символов из символьной строки
StringWriterм	Предназначен для вывода символов в символьную строку

метод Read()

Для чтения одного символа служит метод Read():

```
static int Read()
```

метод Read()

Для чтения одного символа служит метод Read():

```
static int Read()
```

метод ReadLine()

Для считывания строки символов служит метод ReadLine():

```
static string ReadLine()
```

метод Read()

Для чтения одного символа служит метод Read():
```
static int Read()
```

метод ReadLine()

Для считывания строки символов служит метод ReadLine():
```
static string ReadLine()
```

Две формы объявления метода ReadKey().

Для считывания отдельно введенного с клавиатуры символа без построчной буферизации служит метод ReadKey():
```
static ConsoleKeyInfo ReadKey()
static ConsoleKeyInfo ReadKey( bool intercept)
```

метод Read()

Для чтения одного символа служит метод Read():
static int Read()

метод ReadLine()

Для считывания строки символов служит метод ReadLine():
static string ReadLine()

Две формы объявления метода ReadKey().

Для считывания отдельно введенного с клавиатуры символа без построчной буферизации служит метод ReadKey():
static ConsoleKeyInfo ReadKey()
static ConsoleKeyInfo ReadKey(bool intercept)

Замечание

Структура ConsoleKeyInfo состоит из следующих свойств, доступных только для чтения
char KeyChar
ConsoleKey Key
ConsoleModifiers Modifiers

метод Read()

Для чтения одного символа служит метод Read():
```
static int Read()
```

метод ReadLine()

Для считывания строки символов служит метод ReadLine():
```
static string ReadLine()
```

Две формы объявления метода ReadKey().

Для считывания отдельно введенного с клавиатуры символа без построчной буферизации служит метод ReadKey():
```
static ConsoleKeyInfo ReadKey()
static ConsoleKeyInfo ReadKey( bool intercept)
```

Упражнение 7.1

Написать программу для считывания символов введенных с консоли, используя метод ReadKey() и свойства KeyChar, Key и Modifiers

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим)
путь обозначает имя открываемого файла
режим — порядок открытия файла.

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим)
путь обозначает имя открываемого файла
режим — порядок открытия файла.

Таблица 7: Значения из перечисления FileMode

Значение	Описание
FileMode.Append	Добавляет выводимые данные в конец файла
FileMode.Create	Создает новый выходной файл. Существующий файл с таким же именем будет разрушен
FileMode.CreateNew	Создает новый выходной файл. Файл с таким же именем не должен существовать
FileMode.Open	Открывает существующий файл
FileMode.OpenOrCreate	Открывает файл, если он существует. В противном случае создает новый файл
FileMode.Truncate	Открывает существующий файл, но сокращает его длину до нуля

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим – порядок открытия файла.
доступ – указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Пример:

```
// файл открывается только для чтения
FileStream fin = new FileStream("test.dat" , FileMode.Open,
FileAccess.Read);
```

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Метод для чтения одного байта из файла

int ReadByte()
Всякий раз, когда этот метод вызывается, из файла считывается один байт, который затем возвращается в виде целого значения.

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Метод для чтения одного байта из файла

int ReadByte()
Всякий раз, когда этот метод вызывается, из файла считывается один байт, который затем возвращается в виде целого значения.

Метод для чтения блока байтов из файла

int Read(byte[ ] array, int offset, int count)
В данном методе предпринимается попытка считать количество count байтов в массив array, начиная с элемента array[offset]. В случае успеха, возвращается количество байтов считанных из файла.

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Упражнение 7.2

Написать программу для отображения текстового файла. В случае, если файл не удастся открыть, выдать сообщение об ошибке.

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Упражнение 7.2

Написать программу для отображения текстового файла. В случае, если файл не удастся открыть, выдать сообщение об ошибке.

Замечание

По завершении работы с файлом его следует закрыть, вызвав метод Close()

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Упражнение 7.2

Написать программу для отображения текстового файла. В случае, если файл не удастся открыть, выдать сообщение об ошибке.

Упражнение 7.3

Написать программу для записи данных в файл

Распространенный конструктор класса FileStream

FileStream(string путь, FileMode режим, FileAccess доступ)
путь обозначает имя открываемого файла
режим — порядок открытия файла.
доступ — указывается одно из значений FileAccess.Read, FileAccess.Write, FileAccess.ReadWrite

Упражнение 7.2

Написать программу для отображения текстового файла. В случае, если файл не удастся открыть, выдать сообщение об ошибке.

Упражнение 7.3

Написать программу для записи данных в файл

Упражнение 7.4

Написать программу для копирования данных из одного файла в другой

Конструкторы класса StreamWriter

StreamWriter(Stream поток)
StreamWriter(string путь)
StreamWriter(string путь, bool append)
(Если значение параметра append равно true, то выводимые данные присоединяются в конец существующего файла.)

Конструкторы класса StreamWriter

StreamWriter(Stream поток)
StreamWriter(string путь)
StreamWriter(string путь, bool append)
(Если значение параметра append равно true, то выводимые данные присоединяются в конец существующего файла.)

Пример:

```
FileStream file_out;
file_out = new FileStream("test.txt" , FileMode.Create);
StreamWriter file_str_out = new StreamWriter( file_out);
```

Конструкторы класса StreamWriter

StreamWriter(Stream поток)
StreamWriter(string путь)
StreamWriter(string путь, bool append)
(Если значение параметра append равно true, то выводимые данные присоединяются в конец существующего файла.)

Пример:

```
FileStream file_out;
file_out = new FileStream("test.txt" , FileMode.Create);
StreamWriter file_str_out = new StreamWriter( file_out);
```

Упражнение 7.5

Сохранить текст, введенный с клавиатуры, в файл test.txt с помощью символьного потока. Набираемый тест вводится до тех пор, пока в нем не встретится строка "стоп". Использовать различные конструкторы класса StreamWriter

Конструкторы класса StreamReader

StreamReader(Stream поток)
StreamReader(string путь)

Конструкторы класса StreamReader

StreamReader(Stream поток)
StreamReader(string путь)

Замечание

Для обнаружения конца потока в классе StreamReader можно использовать свойство EndOfStream. Это доступное для чтения свойство имеет логическое значение true, когда достигается конец потока, в противном случае — логическое значение false.

Конструкторы класса StreamReader

StreamReader(Stream поток)
StreamReader(string путь)

Замечание

Для обнаружения конца потока в классе StreamReader можно использовать свойство EndOfStream. Это доступное для чтения свойство имеет логическое значение true, когда достигается конец потока, в противном случае — логическое значение false.

Упражнение 7.6

Вывести на экран содержимое текстового файла "Test.txt", созданного в предыдущем примере. Использовать различные конструкторы класса StreamReader

Переадресация стандартных потоков

Для переадресации стандартных потоков в классе Console служат методы SetIn(), SetOut() и SetError():

```
static void SetIn( TextReader новый_поток_ввода)
static void SetOut( TextWriter новый_поток_вывода)
static void SetError( TextWriter новый_поток_сообщений_об_ошибках)
```

Переадресация стандартных потоков

Для переадресации стандартных потоков в классе Console служат методы SetIn(), SetOut() и SetError():

```
static void SetIn( TextReader новый_поток_ввода)
static void SetOut( TextWriter новый_поток_вывода)
static void SetError( TextWriter новый_поток_сообщений_об_ошибках)
```

Упражнение 7.7

Переадресовать вывод потока Console.Out с экрана в файл "logfile.txt".

Переадресация стандартных потоков

Для переадресации стандартных потоков в классе Console служат методы SetIn(), SetOut() и SetError():

```
static void SetIn( TextReader новый_поток_ввода)
static void SetOut( TextWriter новый_поток_вывода)
static void SetError( TextWriter новый_поток_сообщений_об_ошибках)
```

Упражнение 7.7

Переадресовать вывод потока Console.Out с экрана в файл "logfile.txt".

Замечание

При выполнении упражнения 7.7 нужно учитывать, что класс StreamWriter (StreamReader) является производным от класса TextWriter (TextReader).

Стандартный конструктор класса BinaryWriter

BinaryWriter(Stream output)
output обозначает поток, в который выводятся записываемые данные. Для записи в выходной файл в качестве параметра output может быть указан объект, создаваемый средствами класса FileStream.

Таблица 8: Наиболее часто используемые методы класса BinaryWriter

Метод	Описание
void Write(sbyte value)	Записывает значение типа sbyte со знаком
void Write(byte value)	Записывает значение типа byte без знака
void Write(byte[] buffer)	Записывает массив значений типа byte
void Write(short value)	Записывает целочисленное значение типа short (короткое целое)
void Write(ushort value)	Записывает целочисленное значение типа ushort (короткое целое без знака)
void Write(int value)	Записывает целочисленное значение типа
void Write(uint value)	Записывает целочисленное значение типа uint(целое без знака)
void Write(long value)	Записывает целочисленное значение типа long (длинное целое)
void Write(ulong value)	Записывает целочисленное значение типа ulong (длинное целое без знака)
void Write(float value)	Записывает значение типа float (с плавающей точкой одинарной точности)
void Write(double value)	Записывает значение типа double (с плавающей точкой двойной точности)
void Write(decimal value)	Записывает значение типа decimal (с двумя десятичными разрядами после запятой)
void Write(char ch)	Записывает символ
void Write(char[] buffer)	Записывает массив символов
void Write(string value)	Записывает строковое значение типа string, представленное во внутреннем формате с указанием длины строки

Стандартный конструктор класса BinaryReader

BinaryReader(Stream input)
input обозначает поток, из которого вводятся считываемые данные. Для чтения из входного файла в качестве параметра input может быть указан объект, создаваемый средствами класса FileStream.

Таблица 9: Наиболее часто используемые методы класса BinaryReader

Метод	Описание
bool ReadBoolean()	Считывает значение логического типа bool
byte ReadByte()	Считывает значение типа byte
sbyte ReadSByte()	Считывает значение типа sbyte
byte[] ReadBytes(int count)	Считывает количество count байтов и возвращает их в виде массива
char ReadChar()	Считывает значение типа char
char[] ReadChars(int count)	Считывает количество count символов и возвращает их в виде массива
decimal ReadDecimal()	Считывает значение типа decimal
double ReadDouble()	Считывает значение типа double
float ReadSingle()	Считывает значение типа float
short ReadInt16()	Считывает значение типа short
int ReadInt32()	Считывает значение типа int
long ReadInt64()	Считывает значение типа long
ushort ReadUIntl6()	Считывает значение типа ushort
uint ReadUInt32()	Считывает значение типа uint
ulong ReadUInt64()	Считывает значение типа ulong
string ReadString()	Считывает значение типа string, представленное во внутреннем двоичном формате с указанием длины строки. Этот метод следует использовать для считывания строки, которая была записана средствами класса BinaryWriter

Замечание

В классе BinaryReader определены также три приведенных ниже варианта метода Read() и стандартный метод Close().

Замечание

В классе BinaryReader определены также три приведенных ниже варианта метода Read() и стандартный метод Close().

Таблица 10: Варианта метода Read() класса BinaryReader

Метод	Описание
int Read()	Возвращает целочисленное представление следующего доступного символа из вызывающего потока ввода. При обнаружении конца файла возвращает значение -1
int Read(byte[] buffer, int offset, int count)	Делает попытку прочитать количество count байтов в массив buffer, начиная с элемента buffer[offset], и возвращает количество успешно считанных байтов
int Read(char[]buffer, int offset, int count)	Делает попытку прочитать количество count символов в массив buffer, начиная с элемента buffer[offset], и возвращает количество успешно считанных символов

Замечание

В классе BinaryReader определены также три приведенных ниже варианта метода Read() и стандартный метод Close().

Упражнение 7.8

Используя классы BinaryReader и BinaryWriter записать в файл данные разных типов и считать их обратно.

Замечание

В классе BinaryReader определены также три приведенных ниже варианта метода Read() и стандартный метод Close().

Упражнение 7.8

Используя классы BinaryReader и BinaryWriter записать в файл данные разных типов и считать их обратно.

Упражнение 7.9

Использовать классы BinaryReader и BinaryWriter для реализации программы по учету товаров на складе. Вначале вводятся наименования товаров их количество и цена. Данные записываются в файл. Далее по ключевому слову осуществляется поиск в файле, после чего выводится информация о товаре(кол-во на складе, цена за штуку, общая цена всей партии)

Замечание

В классе BinaryReader определены также три приведенных ниже варианта метода Read() и стандартный метод Close().

Упражнение 7.8

Используя классы BinaryReader и BinaryWriter записать в файл данные разных типов и считать их обратно.

Упражнение 7.9

Использовать классы BinaryReader и BinaryWriter для реализации программы по учету товаров на складе. Вначале вводятся наименования товаров их количество и цена. Данные записываются в файл. Далее по ключевому слову осуществляется поиск в файле, после чего выводится информация о товаре(кол-во на складе, цена за штуку, общая цена всей партии)

Замечание

Для проверки достижения конца файла можно использовать исключение EndOfStreamException

Общая форма метода Seek() класса FileStream:

long Seek(long offset, SeekOrigin origin)
offset обозначает новое положение указателя файла в байтах относительно заданного начала отсчета (origin). В качестве origin может быть указано одно из приведенных ниже значений, определяемых в перечислении SeekOrigin

Таблица 11: Перечисление SeekOrigin

Значение	Описание
SeekOrigin.Begin	Поиск от начала файла
SeekOrigin.Current	Поиск от текущего положения
SeekOrigin.End	Поиск от конца файла

Общая форма метода Seek() класса FileStream:

long Seek(long offset, SeekOrigin origin)
offset обозначает новое положение указателя файла в байтах относительно заданного начала отсчета (origin). В качестве origin может быть указано одно из приведенных ниже значений, определяемых в перечислении SeekOrigin

Таблица 11: Перечисление SeekOrigin

Значение	Описание
SeekOrigin.Begin	Поиск от начала файла
SeekOrigin.Current	Поиск от текущего положения
SeekOrigin.End	Поиск от конца файла

Упражнение 7.10

Записать в файл буквы английского алфавита, а затем вывести их на экран в случайном порядке с помощью метода Seek().

Общая форма метода Seek() класса FileStream:

long Seek(long offset, SeekOrigin origin)
offset обозначает новое положение указателя файла в байтах относительно заданного начала отсчета (origin). В качестве origin может быть указано одно из приведенных ниже значений, определяемых в перечислении SeekOrigin

Таблица 11: Перечисление SeekOrigin

Значение	Описание
SeekOrigin.Begin	Поиск от начала файла
SeekOrigin.Current	Поиск от текущего положения
SeekOrigin.End	Поиск от конца файла

Упражнение 7.10

Записать в файл буквы английского алфавита, а затем вывести их на экран в случайном порядке с помощью метода Seek().

Упражнение 7.11

Реализовать программу из предыдущего примера используя свойство Position (см. таблицу 2).

Класс MemoryStream

Для ввода-вывода данных в(из) массив(а) служит класс MemoryStream. В данном классе определено несколько конструкторов. Вот один из них:
MemoryStream(byte[] buffer)

Класс MemoryStream

Для ввода-вывода данных в(из) массив(а) служит класс MemoryStream. В данном классе определено несколько конструкторов. Вот один из них:
MemoryStream(byte[] buffer)

```
Пример: byte[] storage = new byte[255];
// Создать запоминающий поток.
MemoryStream memstrm = new MemoryStream(storage);
```

Класс MemoryStream

Для ввода-вывода данных в(из) массив(а) служит класс MemoryStream. В данном классе определено несколько конструкторов. Вот один из них:
MemoryStream(byte[] buffer)

Пример:
```
byte[] storage = new byte[255];
// Создать запоминающий поток.
MemoryStream memstrm = new MemoryStream(storage);
```

Упражнение 7.12

Заполнить массив storage своими данными и продемонстрировать применение класса MemoryStream. Использовать два подхода: отобразить содержимое массива storage непосредственно и средствами ввода данных из потока (методы класса StreamReader)

Класс MemoryStream

Для ввода-вывода данных в(из) массив(а) служит класс MemoryStream. В данном классе определено несколько конструкторов. Вот один из них:
MemoryStream(byte[] buffer)

Пример:
```
byte[] storage = new byte[255];
// Создать запоминающий поток.
MemoryStream memstrm = new MemoryStream(storage);
```

Упражнение 7.12

Заполнить массив storage своими данными и продемонстрировать применение класса MemoryStream. Использовать два подхода: отобразить содержимое массива storage непосредственно и средствами ввода данных из потока (методы класса StreamReader)

Замечание

Отметим, что данные зачастую записываются в объект назначения не сразу. Вместо этого они буферизуются на уровне операционной системы до тех пор, пока не накопится достаточный объем данных, чтобы записать их сразу одним блоком. Если данные требуется записать без предварительного накопления в буфере, то для этой цели можно вызвать метод Flush().

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Метод Copy()

```
static void Copy (string имя_исходн_файла, string имя_целевого_файла)
static void Copy (string имя_исходн_файла, string имя_целевого_файла,
boolean overwrite)
```

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Метод Copy()

static void Copy (string имя_исходн_файла, string имя_целевого_файла)
static void Copy (string имя_исходн_файла, string имя_целевого_файла, boolean overwrite)

Метод Exists()

Метод Exists() определяет, существует ли файл
static bool Exists(string путь)

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Метод Copy()

static void Copy (string имя_исходн_файла, string имя_целевого_файла)
static void Copy (string имя_исходн_файла, string имя_целевого_файла, boolean overwrite)

Метод Exists()

Метод Exists() определяет, существует ли файл
static bool Exists(string путь)

Метод GetLastAccessTime()

Метод GetLastAccessTime() возвращает дату и время последнего доступа к файлу
static DateTime GetLastAccessTime(string путь)

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Упражнение 7.13

Применить методы Copy(), Exists() и GetLastAccessTime()

Класс File

В классе File имеется ряд удобных методов для чтения из файлов и записи в них, открытия файла и получения ссылки типа FileStream на него. Рассмотрим три из них:

Упражнение 7.13

Применить методы Copy(), Exists() и GetLastAccessTime()

Замечание

Для выполнения нескольких операций над одним и тем же файлом лучше воспользоваться классом FileInfo т.к., в отличии от класса File, он предоставляет методы экземпляра и свойства, а не статические методы.

Метод Parse()

осуществляет преобразование числовых строк в их внутреннее представление.

Метод Parse()

осуществляет преобразование числовых строк в их внутреннее представление.

Таблица 12: Структуры и методы преобразования

Структура	**Метод преобразования**
Decimal	static decimal Parse(string s)
Double	static double Parse(string s)
Single	static float Parse(string s)
Int64	static long Parse(string s)
Int32	static int Parse(string s)
Intl6	static short Parse(string s)
UInt64	static ulong Parse(string s)
UInt32	static uint Parse(string s)
UInt16	static ushort Parse(string s)
Byte	static byte Parse(string s)
Sbyte	static sbyte Parse(string s)

Метод Parse()

осуществляет преобразование числовых строк в их внутреннее представление.

Замечание

Для того чтобы избежать генерирования исключений при преобразовании числовых строк, можно воспользоваться методом TryParse().

Метод Parse()

осуществляет преобразование числовых строк в их внутреннее представление.

Замечание

Для того чтобы избежать генерирования исключений при преобразовании числовых строк, можно воспользоваться методом TryParse().

Пример:

```
static bool TryParse(string s, out int результат)
```