[РАБОТА С ФАЙЛАМИ В С# 1](#_Toc528614851)

[1 Примеры задач на работу с файловыми потоками 1](#_Toc528614852)

[1.1 Работа с потоком байтов 1](#_Toc528614853)

[1.2 Работа с потоком символов 3](#_Toc528614854)

[1.3. Работа с двоичным потоком 4](#_Toc528614855)

[1.4 Асинхронный ввод-вывод 5](#_Toc528614856)

[2. Задания для самостоятельной работы 7](#_Toc528614857)

[ПРИЛОЖЕНИЕ. Справочные данные 10](#_Toc528614858)

# РАБОТА С ФАЙЛАМИ В С#

**Цель работы:** Рассмотреть возможности С# для работы с потоками байтов символов и строк.

**ПОДГОТОВКА К РАБОТЕ:** Изучить методы классов BinaryReader, BinaryWriter, BufferedStream, Directory, Directorylnfo, File, Filelnfo, FileStream, MemoryStream, StreamWriter, StreamReader, StringWriter, StringReader из пространства имен System.IO.

## 1 Примеры задач на работу с файловыми потоками

Рассмотрим работу с файлами в С# на примере приложения, изображенного на рисунке 1.

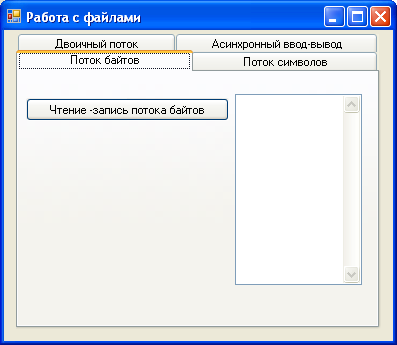


Рис. 1. Пример приложения

### 1.1 Работа с потоком байтов

Для события Click кнопки «Чтение-запись потока байтов» напишем следующий программный код:

private void button1\_Click\_1(object sender, EventArgs e)

{

textBox1.Clear();

//Создание файла

FileStream f = new FileStream("file1.txt", FileMode.Create, FileAccess.ReadWrite);

Random r = new Random();

byte a;

// Запись в файл пятидесяти байт

for (int i = 0; i < 50; ++i)

{

a =(byte) r.Next();

f.WriteByte(a);

}

// установить указатель на начало файла

f.Seek(0, SeekOrigin.Begin);

// получить длину файла в байтах

long j =f.Length;

// чтение и вывод на экран содержимого файла

do

{

a = (byte)f.ReadByte();

textBox1.Text += Convert.ToString(a) + (char)13 + (char)10;

j = j - 1;

} while (j!=0);

//while (f.Length != 0);

f.Close();

}

Текущая позиция в потоке первоначально устанавливается на начало файла (для любого режима открытия, кроме Append) и сдвигается на одну позицию при записи каждого байта.

Для установки желаемой позиции чтения используется метод Seek, имеющий два параметра: первый задает смещение в байтах относительно точки отсчета, зада­ваемой вторым. Точки отсчета задаются константами перечисления SeekOrigin: начало файла - Begin, текущая позиция - Current и конец файла - End.

В данном примере файл создавался в текущем каталоге. Можно указать и пол­ный путь к файлу, при этом удобнее использовать дословные литералы, например:

FileStream f = new FileStream( @"D:\C#\test.txt", FileMode.Create, FileAccess.ReadWrite );

В дословных литералах не требуется дублировать обратную косую черту.

Операции по открытию файлов могут завершиться неудачно, например, при ошиб­ке в имени существующего файла или при отсутствии свободного места на дис­ке, поэтому *рекомендуется всегда контролировать результаты* этих операций.

В случае непредвиденных ситуаций среда выполнения генерирует различные ис­ключения, обработку которых следует предусмотреть в программе, например:

* FileNotFoundException, если файла с указанным именем в указанном каталоге не существует;
* DirectoryNotFoundException, если не существует указанный каталог;
* ArgumentException, если неверно задан режим открытия файла;
* IOException, если файл не открывается из-за ошибок ввода-вывода.

Возможны и другие исключительные ситуации.

Удобно обрабатывать наиболее вероятные ошибки раздельно, чтобы предоста­вить пользователю программы в выводимом сообщении наиболее точную инфор­мацию. В данном примере ошибки не обрабатываются.

При закрытии файла освобождаются все связанные с ним ресурсы, например, для файла, открытого для записи, в файл выгружается содержимое буфера. По­этому рекомендуется всегда закрывать файлы после окончания работы, в осо­бенности файлы, открытые для записи. Если буфер требуется выгрузить, не за­крывая файл, используется метод Flush.

### 1.2 Работа с потоком символов

Окно для работы с потоком символов изображено на рисунке 2.

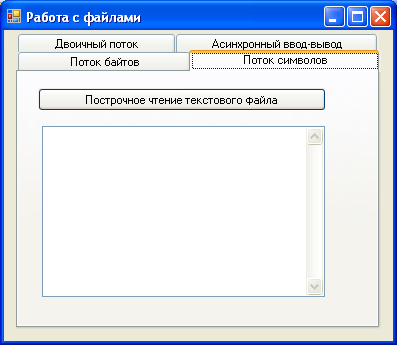


Рис.2. Работа с потоком символов

Для события Click кнопки «Построчное чтение текстового файла» напишем следующий программный код :

private void button2\_Click(object sender, EventArgs e)

{

String s;

textBox2.Clear();

try

{

// Диалог выбора файла

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

// Открытие выбранного файла

StreamReader f = new StreamReader(openFileDialog1.FileName);

// Чтение выбранного файла по строкам

while ((s = f.ReadLine()) != null)

{

textBox2.Text += s + (char)13 + (char)10;

}

f.Close();

}

}

catch (FileNotFoundException e1)

{

textBox2.Text += e1.Message + (char)13 + (char)10;

textBox2.Text += "Проверьте правильность имени файла";

return;

}

catch (Exception e1)

{

textBox2.Text += "Ошибка:" + e1.Message;

}

}

### 1.3. Работа с двоичным потоком

Окно для работы с потоком символов изображено на рисунке 3.

Для события Click кнопки «Формирование двоичного файла» напишем следующий программный код:

private void button3\_Click(object sender, EventArgs e)

{

try

{

BinaryWriter f = new BinaryWriter(new FileStream(@"C:\binary.file", FileMode.Create));

double d = 0;

while (d < 4)

{

f.Write(d);

d += 0.33;

};

f.Seek(16, SeekOrigin.Begin);

f.Write(8888d);

f.Close();

}

catch (Exception e1)

{

textBox2.Text += "Ошибка:" + e1.Message;

return;

}

}

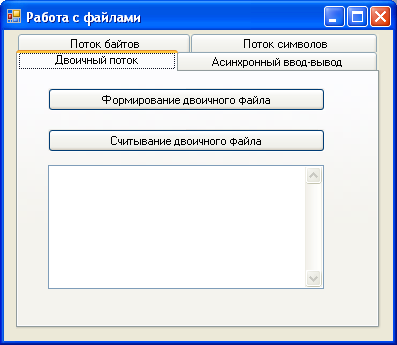


Рис.3. Работа с двоичным потоком

При создании двоичного потока в него передается объект базового потока. При установке указателя текущей позиции в файле учитывается длина каждого зна­чения типа double — 8 байт.

Для события Click кнопки «Считывание двоичного файла» напишем следующий программный код:

private void button4\_Click(object sender, EventArgs e)

{

try

{

FileStream f =new FileStream(@"C:\binary.file", FileMode.Open);

BinaryReader f1 = new BinaryReader(f);

// количество чисел в файле

long n = f.Length / 8;

double[] x = new double[n];

long i = 0;

try

{

while (true) x[i++] = f1.ReadDouble();

}

catch (EndOfStreamException e2) { }

foreach (double d in x) textBox3.Text += Convert.ToString(d) + " ";

f1.Close();

f.Close();

}

catch (FileNotFoundException e1)

{

textBox2.Text += e1.Message + (char)13 + (char)10;

textBox2.Text += "Проверьте правильность имени файла";

return;

}

catch (Exception e1)

{

textBox2.Text += "Ошибка:" + e1.Message;

}

}

Попытка просмотра сформированного программой файла в текстовом редакторе не информативна, поэтому в программе помощью экземпляра BinaryReader считывает содержимое файла в массив вещественных чисел, а затем выводит этот массив на экран. При чтении принимается во внимание тот факт, что метод ReadDouble при обна­ружении конца файла генерирует исключение EndOfStreamException. Поскольку в данном случае это не ошибка, тело обработчика исключений пустое.

### 1.4 Асинхронный ввод-вывод

Окно для реализации асинхронного ввода-вывода изображено на рисунке 4.

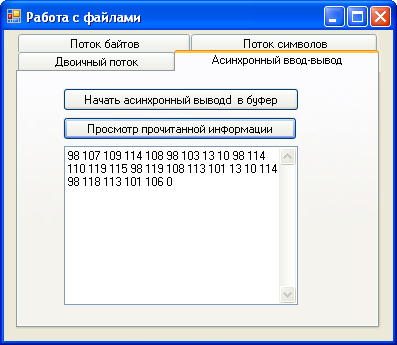
. 

Рис. 4. Асинхронный ввод-вывод

Для удобства восприятия операции чтения из файла и диалога с пользователем оформлены в отдельный класс RWFile:

class RWFile

{

public void OnCompletedRead(IAsyncResult ar) // 1

{

bytes = f.EndRead(ar);

}

public void AsyncRead()

{

f=new FileStream("D:\\BigFile.txt",FileMode.Open,FileAccess.Read,FileShare.Read,buf.Length,true); // 2

callback = new AsyncCallback(OnCompletedRead); // 3

f.BeginRead(buf, 0, buf.Length, callback,null); // 4

}

FileStream f;

public byte[] buf = new byte[77777];

AsyncCallback callback;

public int bytes;

}.

Метод OnCompletedRead (оператор 1) должен получать один параметр стандартно­го типа IAsyncResult, содержащий сведения о завершении операции, которые пе­редаются в метод EndRead.

Файл открывается в асинхронном режиме, об этом говорит значение true по­следнего параметра конструктора (оператор 2). В операторе 3 создается экзем­пляр стандартного делегата AsyncCallback, который инициализируется методом OnCompletedRead.

С помощью этого делегата метод OnCompl etedRead передается в метод BeginRead (опе­ратор 4), который создает отдельный поток, начинает асинхронный ввод и воз­вращает управление в вызвавший поток. Обратный вызов метода OnCompletedRead происходит при завершении операции ввода.

Объявим переменную типа RWFile:

RWFile fw;

Для события Activate формы Form1 напишем следующий программный код:

private void Form1\_Activated(object sender, EventArgs e)

{

fw = new RWFile(); // Созание экземпляра класса

}

Для события Click кнопки «Начать асинхронный вывод в буфер» запишем следующий программный код:

private void button5\_Click(object sender, EventArgs e)

{

fw.AsyncRead();

}

Для события Click кнопки «Просмотр прочитанной информации » запишем следующий программный код:

private void button7\_Click\_1(object sender, EventArgs e)

{

for (int i = 0; i <= fw.bytes; ++i)

{

textBox4.Text += Convert.ToString(fw.buf[i]) + " ";

}

}

## 2. Задания для самостоятельной работы

**Вариант 1.**

1. Сформировать файл последовательности 15 чисел, в кото­рой каждый i*-й* компонент определяется по формуле

y=sin(i\*π / 8), *если i <* 8;

y=4cos(i(π +1)/5), *если i >* 8.

Определить количество положительных значений, содержащихся в сформированном файле.

Вариант 2.

1. Сформировать файл последовательности 20 чисел, в кото­рой каждый *i-й* компонент определяется по формуле

y=sin(i\*π / 8), *если i <* 8;

y=4cos(i(π +1)/5), *если i >* 8.

Определить количество отрицательных значений, содержащихся в сформированном файле.

**Вариант 3.**

1. Сформировать файл из значений случайных величин:

0,324; 0,524; 0,789; 0,556; 0,761; 0,248; 0,345; 0,911; 0,216,

Определить для данной последовательности среднее арифметическое компонентов, значения которых меньше 0,5.

**Вариант 4.**

1. Сформировать файл из значений случайных величин:

0,324; 0,524; 0,789; 0,556; 0,761; 0,248; 0,345; 0,911; 0,216,

Определить для данной последовательности сумму компонентов, значения которых больше 0,5.

**Вариант 5.**

1. Сформировать файл, содержащий фамилии нескольких студентов. Добавить к полученному файлу фамилии еще 2-3 студентов.

**Вариант 6.**

1. Записать в файл оценки (в баллах), полученные некото­рым студентом на экзаменах в течение всех сессий. Добавить в начало файла оценки, полученные на вступительных экзаменах.

**Вариант 7.**

1. Записать в файл оценки (в баллах), полученные некото­рым студентом на экзаменах в течение всех сессий, и определить средний балл.

**Вариант 8.**

1. Сформировать два файла. В один из них поместить фами­лии пяти ваших знакомых, а в другой - номера их телефонов. Составить программу, которая по фамилии вашего знакомого определяет номер его телефона.

**Вариант 9.**

1. Сформировать два файла. В один из них поместить фами­лии пяти ваших знакомых, а в другой - номера их телефонов. Составить программу, которая по номеру телефона вашего знакомого определяет его фамилию.

**Вариант 10.**

1. В текстовом файле, в предложении, содержащем не ме­нее двух слов, поменять местами первое и последнее слово.

**Вариант 11.**

1. Произвести сортировку файла целых чисел методом выбора.

**Вариант 12.**

1. Произвести сортировку файла целых чисел методом пу­зырька.

**Вариант 13.**

1. Перемножить два сверхдлинных целых числа, записан­ных в файле. Результат записать в тот же файл.

**Вариант 14.**

1. В отсортированном файле целых чисел методом бинарного поиска найти заданный элемент и удалить его.

**Вариант 15.**

1. Сформировать файл, компонентами которого являются действительные значения, вычисляемые по формуле ai=(i+1)2sin(iπ/10), где i - номер компонента файла.

Определить, сколько в полученном файле содержится положитель­ных значений.

**Вариант 16.**

1. Сформировать файл, компонентами которого являются действительные значения, вычисляемые по формуле ai=(i+1)2sin(iπ/10), где i - номер компонента файла.

Определить, сколько в полученном файле содержится отрицательных значений.

**Вариант 17.**

1. Сформировать файл целых чисел. Вывести на экран только те компоненты файла, значения которых лежат в интервале от 0 до 25.

**Вариант 18.**

1. Сформировать файл целых чисел. Вывести на экран только четные значения компонентов файла.

**Вариант 19.**

1. Пусть некоторый файл содержит числа Фибоначчи *U0, U1,*..., Un. Получить и дополнить этот файл очередным компонентом *Un+1.*

**Вариант 20.**

1. Сформировать файл, элементами которого являются зна­чения функции *у=* sin(xi) + 2cos(xi) в точках *Х =* (0,1; 0,2; 0,25; 0,33; 1,78; 2,05; 2,23). Определить компонент файла, имеющий минимальное значение.

**Вариант 21.**

1. Произвести сортировку файла целых чисел методом выбора.

**Вариант 22.**

1. Сформировать файл целых чисел. Вывести на экран только нечетные значения компонентов файла.

**Вариант 23.**

1. Записать в файл оценки (в баллах), полученные студен­тами группы по некоторому предмету, и определить средний балл.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как организуется файловый ввод-вывод на основе потоков?
2. Как программируется обработка исключений?
3. Если файл с именем file1.dat не существует, что произойдет при попытке создать объект потока с помощью следующей команды: StreamReader input= new StreamReader (“file1.dat ”):

* Создается и открывается для чтения пустой файл и именем file1.dat;
* Всплывает окно, в котором пользователя просят просмотреть каталоги в поисках требуемого файла и нажать ОК;
* Генерируется исключение FileNotFoundException;
* Генерируется исключение FileLoadException.

1. Если файл с именем secret.dat существует, что произойдет при попытке создать объект потока с помощью следующей команды: StreamWriter out= StreamWriter (“secret.dat ”):

* Содержимое файла secret.dat стирается и поток подготавливается для нового вывода;
* Новый вывод добавляется к концу существующих данных в файле;
* Генерируется исключение FileLoadException;
* Открывается файл secret1.dat.

1. Для посимвольного чтения из файла использовался метод Read. Данный метод возвращает:

* Следующий символ или символ null, если символов больше нет;
* Строку, из которой в дальнейшем можно извлечь символы и null, если символов больше нет;
* Целое число, представляющее следующий символ и 0, если символов больше нет;
* Целое число, представляющее следующий символ и -1, если символов больше нет.

# ПРИЛОЖЕНИЕ. Справочные данные

Табл.1. Основные классы пространства имен System.IO

|  |  |
| --- | --- |
| Класс | Описание |
| BinaryReader, BinaryWriter | Чтение и запись значений простых встроенных типов (целочисленных, логических, строковых и т. п.) во внутренней форме представления |
| BufferedStream | Временное хранение потока байтов (например, для последующего переноса в постоянное хранилище) |
| Directory, Directorylnfo,  File, Filelnfo | Работа с каталогами или физическими файлами: создание, удаление, получение свойств. Возможности классов File и Directory реализованы в основном в виде статических методов. Аналогичные классы Directorylnfo и Filelnfo используют обычные методы |
| FileStream | Произвольный (прямой) доступ к файлу, представленному как поток байтов |
| MemoryStream | Произвольный доступ к потоку байтов в оперативной памяти |
| StreamWriter, StreamReader | Чтение из файла и запись в файл текстовой информации (произвольный доступ не поддерживается) |
| StringWriter, StringReader | Работа с текстовой информацией в оперативной памяти |

Табл.2. Значения перечисления FileAccess

|  |  |
| --- | --- |
| Значение | Описание |
| Read | Открыть файл только для чтения |
| ReadWrite | Открыть файл для чтения и записи |
| Write | Открыть файл только для записи |

Табл.3. 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 |
| Creat | 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 |

Табл.4. Значения перечисления 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 |

Табл. 5. Элементы класса Stream

|  |  |
| --- | --- |
| Элемент | Описание |
| BeginRead, BeginWrite | Начать асинхронный ввод или вывод |
| CanRead, CanSeek, CanWrite | Свойства, определяющие, какие операции поддерживает поток: чтение, прямой доступ и/или запись |
| Close | Закрыть текущий поток и освободить связанные с ним ресурсы (сокеты, указатели на файлы и т. п.) |
| EndRead, EndWrite | Ожидать завершения асинхронного ввода; закончить асинхронный вывод |
| Flush | Записать данные из буфера в связанный с потоком источник данных и очистить буфер. Если для данного потока буфер не используется, то этот метод ничего не делает |
| Length | Возвратить длину потока в байтах |
| Position | Возвратить текущую позицию в потоке |
| 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 |
| SetLength | Установить длину текущего потока |
| Write, WriteByte | Записать последовательность байтов (или один байт) в текущий поток и переместить указатель в потоке на количество записанных байтов |

Табл.6. Наиболее важные элементы базового класса TextWriter

|  |  |
| --- | --- |
| Элемент | Описание |
| Close | Закрыть файл и освободить связанные с ним ресурсы. Если в процессе записи используется буфер, он будет автоматически очищен |
| Flush | Очистить все буферы для текущего файла и записать накопленные в них данные в место их постоянного хранения. Сам файл при этом не закрывается |
| NewLine | Используется для задания последовательности символов, означающих начало новой строки. По умолчанию используется последовательность «возврат каретки» — «перевод строки» (\r\n) |
| Write | Записать фрагмент текста в потоr |
| WriteLine | Записать строку в поток и перейти на другую строку |

Табл.7. Наиболее важные элементы класса TextReader

|  |  |
| --- | --- |
| Элемент | Описание |
| Peek | Возвратить следующий символ, не изменяя позицию указателя в файле |
| Read | Считать данные из входного потока |
| ReadBl ock | Считать из входного потока указанное пользователем количество символов и записать их в буфер, начиная с заданной позиции |
| ReadLine | Счмтать 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 |

Табл. 8. Наиболее важные элементы класса BinaryWriter

|  |  |
| --- | --- |
| Элемент | Описание |
| BaseStream | Базовый поток, с которым работает объект BinaryWriter |
| Close | Закрыть поток |
| Flush | Очистить буфер |
| Seek | Установить позицию в текущем потоке |
| Wri te | Записать значение в текущий поток |

Табл.9. Наиболее важные элементы класса BinaryReader

|  |  |
| --- | --- |
| Элемент | Описание |
| BaseStream | Базовый поток, с которым работает объект BinaryReader |
| Close | Закрыть поток |
| PeekChar | Возвратить следующий символ без перемещения внутреннего указателя в потоке |
| Read | Считать поток байтов или символов и сохранить в массиве, передаваемом как входной параметр |
| ReadXXXX | Считать из потока данные определенного типа (например, ReadBoolean, ReadByte, Readlnt32 и т. д.) |