Рязанский станкостроительный колледж РГРТУ

Основы алгоритмизации и программирования

Тема 4. Структурированные типы данных. Файлы.

Рязань 2022

Оглавление

Файлы

Общая информация

Файл – одна из наиболее фундаментальных структур данных. Для управления множеством файлов в состав операционной системы входит файловая система, определяющая, каким образом файлы именуются и где они размещаются.

Фреймворк .NET предоставляет большие возможности по управлению и манипуляции файлами и каталогами, которые по большей части сосредоточены в **пространстве имен System.IO**. Соответственно для использования классов для работы с файлами нужно описать пространство имен **uses System.IO.**

Для работы с дисками имеется класс ***DriveInfo***.

Для работы с каталогами предназначены сразу два класса: ***Directory*** и ***DirectoryInfo.***

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.

Класс ***FileStream*** представляет возможности по считыванию из файла и записи в файл. Он позволяет работать как с текстовыми файлами, так и с бинарными.

Для работы с текстовыми файлами определены специальные классы: ***StreamReader*** и ***StreamWriter***.

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

Работа с файлами может осуществляться в режиме последовательного доступа, т.е. информация последовательно записывается в файл и также последовательно считывается из файла.

Работа с файлами может осуществляться в режиме произвольного доступа, т.е. информация может записываться в произвольном порядке и также произвольно считывается из файла.

Файл может быть открыт:

- для чтения (входной файл);
- для записи (выходной файл);
- для чтения и записи (двунаправленный обмен).

Текстовые файлы

Текстовый файл является файлом последовательного доступа, т.е. информация последовательно записывается в файл и также последовательно считывается из файла.

Текстовые файлы являются форматируемыми. Операции обмена с текстовым файлом сопровождаются преобразованием информации аналогично тому, как это происходит для стандартных операций.

При выводе чисел в текстовый файл, они преобразуются из внутреннего числового представления во внешнее текстовое представление.

При чтении из файла чисел необходимо преобразование из текстового представления во внутреннее числовое представление.

Если содержимое текстового файла обрабатывается строки, преобразование не выполняется.

Хотя текстовые файлы полезны во многих случаях и содержат удобный для восприятия человека текст, у них нет гибкости двоичных файлов, содержащих исходные двоичные данные во внутреннем представлении, непосредственно используемом программой.

Описание классов для работы с текстовыми файлами

Для работы с текстовыми классами определены специальные классы: ***StreamReader*** и ***StreamWriter***.

Запись в файл и StreamWriter

Для записи в текстовый файл используется класс ***StreamWriter***. Некоторые из его конструкторов, которые могут применяться для создания объекта ***StreamWriter***:

StreamWriter(string path)	Создает экземпляр класса, через параметр ***path*** передается путь к файлу, который будет связан с потоком. Открывает новый файл, если файл существовал, он перезаписывается.
StreamWriter(string path, bool append)	Создает экземпляр класса, через параметр ***path*** передается путь к файлу, который будет связан с потоком. Параметр ***append*** указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно ***true***, то новые данные добавляются в конец файла. Если равно ***false***, то файл перезаписывается заново.

Свою функциональность ***StreamWriter*** реализует через следующие методы:

int Close()	Закрывает записываемый файл и освобождает все ресурсы.
void Flush()	Записывает в файл оставшиеся в буфере данные и очищает буфер.
void Write(string value)	Записывает в файл данные простейших типов, как int, double, char, string и т.д.
void WriteLine(string value)	Также записывает данные, только после записи добавляет в файл символ окончания строки.

Чтение из файла и StreamReader

Класс ***StreamReader*** позволяет нам легко считывать весь текст или отдельные строки из текстового файла.

Некоторые из конструкторов класса StreamReader:

StreamReader(string path)	Создает экземпляр класса, через параметр path передается путь к считываемому файлу. Файл открывается для чтения.

Среди методов StreamReader можно выделить следующие:

void Close()	Закрывает считываемый файл и освобождает все ресурсы
int Peek()	Возвращает следующий доступный символ, если символов больше нет, то возвращает -1
int Read()	Считывает следующий символ в численном представлении, если больше нет доступных символов, возвращает значение -1.
string ReadLine()	Считывает следующую строку в файле, если достигнут конец файла, возвращает значение ***null***.
string ReadToEnd()	Считывает весь текст из файла.

Алгоритм записи информации в текстовый файл

1. Создаем поток для работы с файлом.

Для каждого файла создаем отдельный поток. Поток определяется как объект соответствующего класса.

Пример:

```
StreamWriter outFile;//определение потоковой переменной outFile для вывода
```

2. Связывем поток с файлом и открываем его для записи.

Пример:

```
outFile = new StreamWriter("rez.txt");//Открываем файл для записи
```

3. Запись информации в файл.

После того, как файл открыт, можно считать из него или записать в него данные.

Для текстового файла запись осуществляются с помощью методов ***Write*** и ***WriteLine*** так же, как это делалось для консольного вывода.

Пример:

```
int value = 5;
outFile.WriteLine(value);;
```

4. Закрытие файла и уничтожение потока

Для явного закрытия файла используется функция ***close()*** соответствующего потокового класса, не имеющая параметров и возвращаемого значения. Она чистит буфер потока, отсоединяет поток от файла и закрывает файл. При этом потоковая переменная продолжает существовать и может быть связана с другим файлом.

Пример:

```
outFile.Close();//Закрываем файл
```

Алгоритм чтения информации из текстового файла

1. Создаем поток для работы с файлом.

Для каждого файла создаем отдельный поток. Поток определяется как объект соответствующего класса.

Пример:

```
StreamReader inFile;//Определение потоковой переменной inFile для ввода
```

2. Связываем поток с файлом и открываем его для чтения.

Пример:

```
inFile = new StreamReader("rez.txt");//Открываем файл для чтения
```

3. Чтение информации из файла.

После того, как файл открыт, можно считать из него или записать в него данные.

Для текстового файла запись осуществляются с помощью методов ***Read*** и ***ReadLine*** так же, как это делалось для консольного вывода.

Пример:

```
int value;
value = Convert.ToInt32(inFile.ReadLine());
```

4. Закрытие файла и уничтожение потока

Для явного закрытия файла используется функция ***close()*** соответствующего потокового класса, не имеющая параметров и возвращаемого значения. Она чистит буфер потока, отсоединяет поток от файла и закрывает файл. При этом потоковая переменная продолжает существовать и может быть связана с другим файлом.

При завершении программы происходит закрытие файла (автоматически вызывается функция ***close()***) и уничтожение потоковой переменной.

Пример:

```
inFile.Close();//Закрываем файл
```

Пример 1

Составить программу, которая записывает в текстовый файл заданное количество случайных значений, далее считывает их из файла и считает среднее арифметическое и сумму.

Методика решения задачи

С клавиатуры вводим количество значений. Формируем случайное значение и записываем его в файл, повторяя эту процедуру с помощью цикла for заданное количество раз. Далее т.к. знаем количество записей в файле, в цикле for считываем значения из файла и суммируем их. В конечном итоге рассчитываем среднее и выводим пользователю результат на экран.

Код программы

```
StreamReader inFile;//Определение потоковой переменной inFile для ввода
StreamWriter outFile;//Определение потоковой переменной outFile для вывода
int x,  //Текущее число
    num,//Количество чисел
    i,  //Счетчик - индекс
    sum;//Сумма
Console.Write("Введите количество чисел=");
num = Convert.ToInt32(Console.ReadLine());

//Заполняем файл случайными значениями
Random rnd = new Random();
outFile = new StreamWriter("rez.txt");//Открываем файл для записи
for (i = 1; i <= num; i++)
    outFile.WriteLine(rnd.Next(200)); //Записываем числа в файл
outFile.Close();//Закрываем файл

//Читаем значения из файла и суммируем их
inFile = new StreamReader("rez.txt");//Открываем файл для чтения
sum = 0;
for (i = 1; i <= num; i++)
{
    x = Convert.ToInt32(inFile.ReadLine()); //Читаем числа из файла
    sum = sum + x;//Складываем числа
    Console.Write("{0,5}", x);
}
inFile.Close();//Закрываем файл

//Вывод ответа
Console.WriteLine();
Console.WriteLine("Сумма чисел = " + sum);
Console.WriteLine("Среднее = " + (double)sum / num);
Console.ReadKey();//Пауза
```

Пример 2

Заполнить текстовый файл целыми числовыми значениями, которые расположить каждое значение на новой строке. Считать числа из файла и подсчитать сумму.

Методика решения задачи

Для работы программы формируем текстовый файл с целыми числовыми значениями, для чего можно использовать программу Блокнот. Каждое значение располагаем на отдельной строке.

Т.к. количество записей в файле неизвестно, то считываем значения из файла, пока они реально не закончатся.

Код программы

```
//31.03.2022 Сидоров С.С. ИСП-21
//Пример2. Заполнить текстовый файл целыми числовыми значениями,
//которые расположить каждое значение на новой строке. Считать
//числа из файла и подсчитать сумму.

StreamReader inFile;//Определение потоковой переменной inFile для ввода
int value,  //Текущее число
    i,  //Счетчик - индекс
    sum;//Сумма
string line;//Считываемая строка
string fileName; //Имя файла
Console.Write("Введите имя файла = ");
fileName = Console.ReadLine();
inFile = new StreamReader(fileName);//Открываем файл для чтения
sum = 0;
//Считываем строки, пока не встретим пустую строку
while ((line = inFile.ReadLine()) != null)
{
    value = Convert.ToInt32(line); //Берем считанное значение
    sum = sum + value;//Складываем числа
    Console.Write(value + " ");
}
Console.WriteLine();
Console.WriteLine("Сумма чисел = " + sum);
inFile.Close();//Закрываем файл
Console.ReadKey();//Пауза
```

Практическая работа №13.

Файлы

Составьте алгоритм и программу обработки файлов в соответствии с вариантом задания.

Варианты заданий

1. Записать 200 случайных чисел в файл a.txt. В файл b.txt вывести четные, а в файл c.txt нечетные.
2. Составить программу, которая считывает с клавиатуры числа и записывает их в файл. Считывать числа до тех пор, пока не введено число 0. Далее считать числа из файла и рассчитать сумму чисел.
3. Считать из простого текстового файла символы и установить количество букв «а» и «о». Текстовый файл сформировать вручную (блокнот, far manager).
4. Составить программу, которая записывает в два файла заданное количество случайных значений, далее считывает их из файлов и считает сумму эквивалентных значений из первого и второго файла и результат записывает третий файл.
5. Записать 1000 случайных чисел в диапазоне [-100;100] в файл a.txt. В файл otr.txt вывести отрицательные значения, в файл plus.txt положительные значения, в файл even.txt все нечетные числа.
6. Составить программу, которая считывает с клавиатуры числа и четные записывает в файл even.txt, нечетные записывает в файл odd.txt. Считывать числа до тех пор, пока не введено число 0. Далее рассчитать сумму нечетных чисел и произведение четных чисел.
7. Сформировать вручную текстовый файл (блокнот, far manager), который содержит латинский алфавит. Считать из файла посимвольно строку текста и вывести коды символов латинского алфавита.
8. Дан файл целых чисел. Создать новый файл из него, где все элементы с четными номерами.
9. Записать 150 случайных чисел в файл a.txt. В файл b.txt вывести четные, а в файл c.txt кратные 5.
10. Составить программу, которая считывает с клавиатуры числа и записывает их в файл. Считывать числа до тех пор, пока не введено число 0. Далее считать числа из файла и рассчитать среднее всех чисел.
11. Считать из простого текстового файла символы и установить количество букв «б» и «г». Текстовый файл сформировать вручную (блокнот, far manager).
12. Составить программу, которая записывает в два файла заданное количество случайных значений, далее считывает их из файлов и считает сумму эквивалентных значений из первого и второго файла, если значения из первого файла больше значений из второго файла и результат записывает третий файл.
13. Записать 1000 случайных чисел в диапазоне [-50;100] в файл a.txt. В файл otr.txt вывести отрицательные значения, в файл plus.txt положительные значения, в файл dip.txt значения из диапазона [-5;50].
14. Составить программу, которая считывает с клавиатуры положительные и отрицательные числа и четные записывает в файл even.txt, нечетные записывает в файл odd.txt. Считывать числа до тех пор, пока не введено число 0. Далее рассчитать сумму нечетных отрицательных чисел и произведение четных положительных чисел.
15. Сформировать вручную текстовый файл (блокнот, far manager), который содержит русский алфавит. Считать из файла посимвольно строку текста и вывести коды символов русского алфавита.

16. Даны два файла с одинаковым количеством целых чисел. Определить сколько соответствующих элементов файлов совпадают. Текстовые файлы сформировать вручную (блокнот, far manager).

17. Дан файл целых чисел. Определить наибольшее из них, принадлежащее интервалу [a;b]. Текстовый файл сформировать вручную (блокнот, far manager).

18. Дан файл целых чисел. Определить сколько раз в нем повторяется максимальное значение. Текстовый файл сформировать вручную (блокнот, far manager).

Контрольные вопросы

1. Понятие файла. Прочитайте полное имя (спецификацию) файла: **A:\WORK\text1.txt**
2. Как выбираются элементы файла?
3. Чем отличается прямой доступ к файлу от последовательного доступа?
4. Как описывается текстовый файл?
5. Чем отличаются типизированные файлы от нетипизированных?
6. Какие действия выполняются при вводе числовых данных из текстового файла?