# **Практическая работа № 10.** Работа с потоками и файловой системой в С#. Чтение и запись текстовых файлов. Ввод и вывод табличных данных

Объём учебного времени – 2ч

Методические рекомендации

1. Цель работы: Научиться создавать приложения для чтения и записи бинарных файлов, XML-документов.
2. Перечень необходимых средств обучения:
   1. Технические средства обучения:

Компьютер Core i3 3.0, 2 Gb оперативной памяти, винчестер 250 Gb, DVD

* 1. Программное обеспечение:
* ОС Windows 7
* MS Visual Studio 2010

1. Основные теоретические положения:

**Массивы C#**

Массив представляет собой совокупность переменных одного типа с общим для обращения к ним именем. В C# массивы могут быть как одномерными, так и многомерными. Массивы служат самым разным целям, поскольку они предоставляют удобные средства для объединения связанных вместе переменных.

Массивами в C# можно пользоваться практически так же, как и в других языках программирования. Тем не менее у них имеется одна особенность: они реализованы в виде объектов.

Для тoго чтобы воспользоваться массивом в программе, требуется двухэтапная процедура, поскольку в C# массивы реализованы в виде объектов.

Во-первых, необходимо объявить переменную, которая может обращаться к массиву.

Во-вторых, нужно создать экземпляр массива, используя оператор new.

// Объявляем массив

int[] myArr = new int[5];

// Инициализируем каждый элемент массива вручную

myArr[0] = 100;

myArr[1] = 23;

myArr[2] = 25;

myArr[3] = 31;

myArr[4] = 1;

foreach (int i in myArr) {} //обращаемся к массиву

Синтаксис инициализации массива с использованием ключевого слова new

int[] myArr = new int[] {10,20,30,40,50};

Синтаксис инициализации массива без использования ключевого слова new

string[] info = { "Фамилия", "Имя", "Отчество" };

// Используем ключевое слово new и желаемый размер

char[] symbol = new char[4] { 'X','Y','Z','M' };

Ключевое слово var позволяет определить переменную так, чтобы лежащий в ее основе тип выводился компилятором. Аналогичным образом можно также определять неявно типизированные локальные массивы. С использованием такого подхода можно определить новую переменную массива без указания типа элементов, содержащихся в массиве.

Свойство Length

Реализация в C# массивов в виде объектов дает целый ряд преимуществ. Одно из них заключается в том, что с каждым массивом связано свойство Length, содержащее число элементов, из которых может состоять массив. Следовательно, у каждого массива имеется специальное свойство, позволяющее определить его длину.

**Управление файловой системой**

Классы, которые используются для просмотра файловой системы и выполнения таких операций, как перемещение, копирование и удаление файлов, показаны на следующей диаграмме. Пространство имен каждого класса показано в скобках под именем каждого класса на диаграмме:

Назначение классов:

* FileSystemInfo — базовый класс, который представляет любой объект файловой системы.
* FileInfo и File — представляют файл в файловой системе.
* DirectoryInfo и Directory — представляют папку в файловой системе.
* Path — содержит статические члены, которые можно использовать для манипуляций именами путей доступа.

В Windows объекты, которые содержат файлы и используются для организации файловой системы, называются папками. Например, в пути доступа

C:\My Documents\ReadMe.txt файлом является ReadMe.txt , а My Documents  — папкой.

**Папка** (folder) является специфическим для Windows терминам: практически во всех других операционных системах вместо папки используется термин каталог (directory). В соответствии с желанием Microsoft, чтобы .NET была максимально независимой от операционной системы, соответствующие базовые классы .NET называются Directory и DirectoryInfo

**Прочитать текстовый файл**

В следующем коде чтение текстового файла. Дан текстовый файл с именем Sample.txt. Для работы с файлами требуется добавить следующий код в начало файла Class1.cs: **using System.IO;** Метод **Main**:

String line;

try

{

StreamReader sr = new StreamReader("C:\\Sample.txt");//Путь к файлу

line = sr.ReadLine();//Читаем первые строки файла

while (line != null) //Читаем, пока не достигнем конца файла

{

Console.WriteLine(line); //Пишем в окно консоли

line = sr.ReadLine();//Читаем новую линию в файле

}

sr.Close();//Закрываем файл

Console.ReadLine();

}

catch(Exception e)

{

Console.WriteLine("Exception: " + e.Message);

}

finally

{

Console.WriteLine("Executing finally block.");

}

Пример 2. **Запись в текстовый файл**. Используется класс **StreamWriter**, чтобы открыть, записать и закрыть текстовый файл. Как и в классе **StreamReader**, путь к текстовому файлу можно передать конструктору **StreamWriter**, чтобы автоматически открыть файл. Метод **WriteLine** записывает полные строки текста в текстовый файл.

try

{

StreamWriter sw = new StreamWriter("C:\\Test.txt");//Имя файла

sw.WriteLine("Hello World!!"); //Запись строки в файл

sw.WriteLine("From the StreamWriter class");//Запись следующей линии в файл

sw.Close();//Закрытие файла

}

catch(Exception e)

{

Console.WriteLine("Exception: " + e.Message);

}

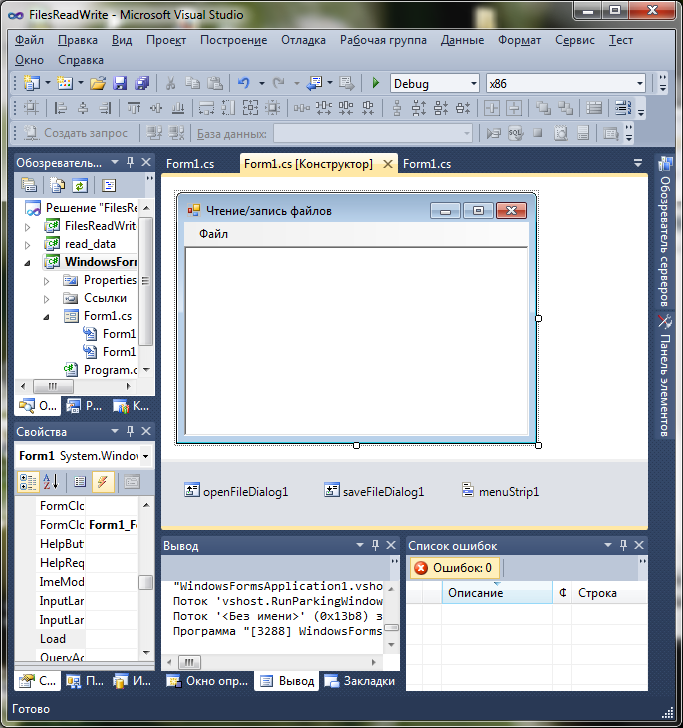
finally

{

Console.WriteLine("Executing finally block.");

}

1. Практические указания:
   1. **Создание программы для чтения/записи текстового файла в кодировке Windows 1251**
      1. Создайте новое Windows приложение C#.
      2. На форме разместите следующие компоненты:



* + 1. Измените следующие свойства добавленных компонентов:
       - Для компонента richTextBox1 свойство Multiline установите на true;
       - Заголовок формы измените на "Чтение/запись файлов";
       - Для компонента openFileDialog1 свойство FileName установите в @"E:\Text.txt";
       - Для компонентов openFileDialog1 и saveFileDialog1 свойство Filter установите "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";
    2. В меню Файл расположите команды: Открыть, Сохранить как и Выход.
    3. Напишите процедуру для выбора команды **Открыть** в меню:
       - Показать окно **openFileDialog1** с помощью метода **ShowDialog**();
       - Если введённое имя файла равно String.Empty, то return;
       - Напишите инструкцию:

**try**

**{**

**var rd = new System.IO.StreamReader(openFileDialog1.FileName, Encoding.GetEncoding(1251));**

**richTextBox1.Text = rd.ReadToEnd();**

**rd.Close();**

**}**

**catch (System.IO.FileLoadException s)**

**{**

**MessageBox.Show(s.Message + "\nНет такого файла", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);**

**}**

**catch (Exception s)**

**{**

**MessageBox.Show(s.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);**

**}**

* + 1. Создадим процедуру zap() для потоковой записи в файл:

**private void zap()**

**{**

**try**

**{**

**var wr = new System.IO.StreamWriter(saveFileDialog1.FileName, false, System.Text.Encoding.GetEncoding(1251));**

**wr.Write(richTextBox1.Text);**

**wr.Close();**

**richTextBox1.Modified = false;**

**}**

**catch (Exception s)**

**{**

**MessageBox.Show(s.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);**

**}**

**}**

* + 1. Напишите инструкцию для Сохранить:

**saveFileDialog1.FileName = openFileDialog1.FileName;**

**if (saveFileDialog1.ShowDialog() == DialogResult.OK)**

**zap();**

* + 1. Напишите процедуру для команды Закрыть.
    2. На событие FormClosing для формы напишите процедуру:
       - Если richTextBox1 не изменялся, то return;
       - Опишем переменную MBox = MessageBox.Show("Текст был изменён." + " Сохранить изменения?", "Редактор", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Exclamation);
       - Если MBox равна DialogResult.No (нажата клавиша No), то return;
       - Если выбрана клавиша Cancel, то **e.Cancel = true**;
       - Если нажата клавиша **Yes**, то:

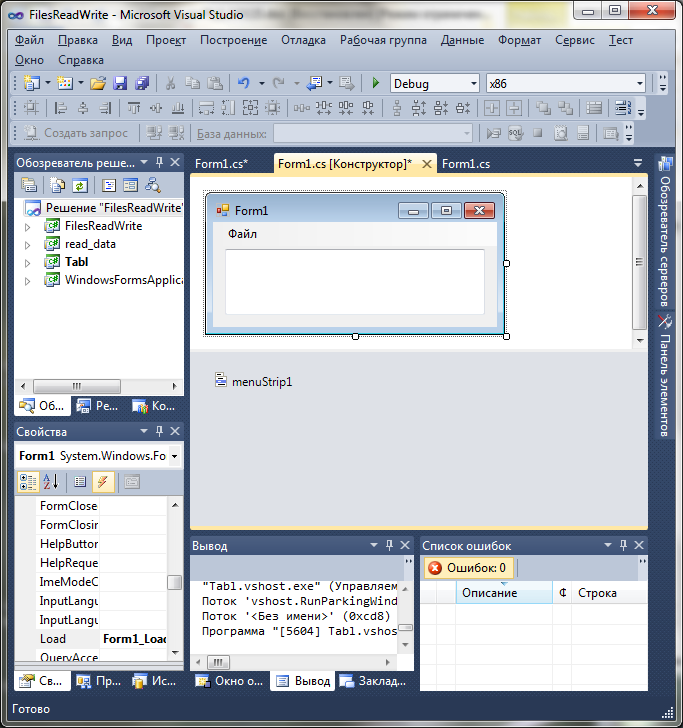
**{**

**if (saveFileDialog1.ShowDialog() == DialogResult.OK) { zap(); return; }**

**else e.Cancel = true;**

**}**

* + 1. Самостоятельно реализуйте добавление пункта меню Правка (Копировать, Вырезать, Вставить). Указания для выполнения: Для решения задачи требуется использовать свойство **SelectedText** для окна ввода, а также объект Clipboard и его методы **SetDataObject**, **GetDataObject**.
  1. **Ввод-вывод табличных данных**
     1. Вызовите контекстное меню вашего решения и выберите команду – Добавить – Создать проект – Windows приложение.
     2. Расположите на форме элементы меню, текстовое поле **textBox1**:



* + 1. Измените свойства:
       - Формы – размер 438\*272, заголовок Таблица телефонов.
       - Окна ввода – Multiline = true, размер 415\*229, ScrollBars = ScrollBars.Vertical, Font Courier New, размер 9.
    2. В меню добавьте команды: Показать таблицу в Блокноте и Выход.
    3. На событие загрузки формы напишите:
       - Опишите строковую константу HC = "\r\n" для формирования новой строки;
       - Опишите строковый массив Names, который содержит имена из вашей телефонной книжки (не менее 5 имен); String[] Names ={"",…}
       - Опишите строковый массив Telephons, который содержит телефоны, в соответствии с первым массивом имён;
       - Опишите переменную i и инициализируйте её 0;
       - Организуйте массив foreach для просмотра в цикле имён и вывода в окно вывода textBox1 значений:

**foreach (var n in Names)**

**{**

**textBox1.Text += String.Format("{0,-21}{1,-21}" + HC, n, Telephons[i]);**

**i++;**

**}**

* + - * Опишите переменную wr для формирования потока для записи данных в файл @"E:\Table.txt";
      * Запишите данные из textBox1.Text в поток записи;
      * Закройте поток.
    1. Для реализации пункта меню **Показать таблицу в Блокноте**, напишите процедуру:

**try**

**{**

**System.Diagnostics.Process.Start("Notepad", @"E:\Table.txt");**

**}**

**catch (Exception situation)**

**{**

**MessageBox.Show(situation.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);**

**}**

* + 1. Напишите процедуру для закрытия формы.
    2. Самостоятельно реализуйте вывод таблицы данных в Internet Explorer. Указания для выполнения: на основании двух данных массивов должна сформировать таблица в формате HTML, создан файл с расширением html, а затем файл должен быть открыт браузером, с помощью метода **System.Diagnostics.Process.Start("Iexplorer", @"E:\Table.html");**
    3. Самостоятельно разберитесь с возможностью формирования таблицы с помощью элемента управления **DataGridView**. Указания для выполнения: для формирования таблицы требуется создать новый экземпляр класса DataTable -tabl, затем заполнить свойства Columns с помощью метода Add – добавить два поля заголовка таблицы (Имя, Телефоны), затем в цикле добавить **tabl.Rows.Add(Names[i],Telephons[i]).** Затем свойству DataSource компонента dataGridView1 присвоить значение tabl.

1. Содержание отчёта по практической работе
   1. Цель работы.
   2. Результаты выполнения работы;
   3. Выполненные работы в электронном виде представляются на сетевом диске H (X)
2. Контрольные вопросы
   1. Организация ввода и вывода данных.
   2. Использование **String.Format**.
   3. Возможности открытия файлами приложениями операционной системы.
   4. Элемент управления **DataGridView**.
3. Список рекомендуемой литературы:
   1. Агальцов В.П. Базы данных. Распределённые и удалённые базы данных. Учебник М.: ИД «Форум»: ИНФРА-М, 2011. - 272с.
   2. Голицына О.Л. Системы управления базами данных: учеб. пособие.- М.:ФОРУМ: ИНФРА-М,2006.- 432с
   3. Голощапов А.Л. Microsoft Visual Studio 2010, - СПб.: БХВ-Петербург, 2011. – 544 с.
   4. Грекул В.И. Проектирование информационных систем. Курс лекций. Учебное пособие. М.: Интернет Ун-т Информ. технологий- 2005, 304с.
   5. Зиборов В.В. Visual C# 2012 на примерах. – СПб.: БХВ-Петербург, 2013. – 480 с.
   6. Культин Н.Б. Microsoft Visual C# в задачах и примерах. – СПб.: БХВ-Петербург, 2013. – 320 с.
   7. Петров В.Н. Избачков Ю.С., Информационные системы. СПб.: Питер, 2006, 656с.
   8. Савицкий Н.М. Технологии организации, хранения и обработки данных: учеб. пособие.- М.:ФОРУМ: ИНФРА-М,2001.- 232с.
   9. Флеонов М.Е. Библия С#. – 2-е изд., перераб. и доп. СПб.: БХВ-Петербург, 2011. – 560 с.