Модель разработки приложений в .NET

РАБОТА С ФАЙЛАМИ

- Чтение и запись в файлы. Классы **StreamReader** и **StreamWriter**
- Классы **DirectoryInfo**, **FileInfo**
- Кодировка файлов. Класс **Encoding**

МАССИВЫ В C#

- Одномерные массивы
- Многомерные (n-мерные) массивы
- Зубчатые (рваные) массивы

Работа с файлами

Сборка **System.IO**

Класс StreamReader – служит для чтения строк из текстового файла в определенной кодировке.

StreamReader(fileName) – новый экземпляр класса для указанного имени файла.

StreamReader(fileName, Encoding) – новый экземпляр класса для указанного имени файла с заданной кодировкой символов.

Основные методы класса StreamReader

Peek() – возвращает следующий доступный символ, но не использует его.
Close() – закрывает объект StreamReader и основной поток и освобождает системные ресурсы, связанные с устройством чтения.
Read() – выполняет чтение следующего символа из входного потока и перемещает положение символа на одну позицию вперед.
ReadBlock() – считывает указанное максимальное количество символов из текущего потока и записывает данные в буфер, начиная с заданного индекса.
ReadLine() – выполняет чтение строки символов из текущего потока и возвращает данные в виде строки.
ReadToEnd() – считывает все символы, начиная с текущей позиции до конца потока.

Основные методы класса

StreamReader

```
StreamReader  sr = new StreamReader(path);
   string str = sr.ReadToEnd();
sr.Close();
```

```
StreamReader  sr  =  new  StreamReader( path );
while (sr.Peek()  > -1)
{
      Console.WriteLine(sr.ReadLine() );
}
sr.Close();
```

Работа с файлами

Класс **StreamWriter** – служит для записи символов в текстовый файл в определенной кодировке.

Конструктор:
StreamWriter(fileName, Boolean, Encoding)

fileName – имя файла
Boolean – указывает на перезапись файла, либо в него могут быть добавлены данные (True – добавляет записи, False – перезаписывает)
Encoding – кодировка файла

Основные методы класса StreamWriter

Close() – закрывает объект StreamWriter и базовый поток.
Flush() – очищает все буферы для текущего средства записи и вызывает запись всех данных буфера в основной поток.
Write() – записывает в текстовую строку или поток указанное в параметрах значение.
WriteLine() – записывает в текстовую строку или поток строку вместе с признаком конца строки.

```
StreamWriter  sw = new  StreamWriter(“text.txt”);
foreach  (DirectoryInfo  dir  in cDirs)
{
      sw.WriteLine(dir.Name);
}
sw.Flush();
sw.Close();
```

Кодировка файлов

Класс **Encoding** – представляет кодировку символов.

```
Encoding  enc = new Encoding();
```

ASCII – кодировка для набора символов ASCII (7-разрядных).
Default – кодировка для текущей кодовой страницы ANSI операционной системы.
Unicode – кодировка для формата UTF-16 с прямым порядком байтов.
UTF32, UTF7, UTF8 – кодировка для соответствующих форматов.
GetEncoding( ) – кодировка с указанными параметрами.

Кодировка файлов

Пример:

```
//считываем текст из файла (кодировка 866)
StreamReader  sr = new StreamReader("C:\\old_text.txt",
    Encoding.GetEncoding(866));
string  text = sr.ReadToEnd();
String ntxt = text.Replace(“old”, “new”);
sr.Close();

//записываем текст в другой файл (с кодировкой 1251)
StreamWriter  sw = new StreamWriter("C:\\new_text.txt", false,
Encoding.GetEncoding(1251));
sw.Write(ntxt);

sw.Flush();
sw.Close();
```

Классы: DirectoryInfo, FileInfo

DirectoryInfo – предоставляет методы экземпляра класса для создания, перемещения и перечисления в каталогах и подкаталогах.

```
DirectoryInfo di = new DirectoryInfo(string  patch);
```

FileInfo – предоставляет свойства и методы экземпляра для создания, копирования, удаления, перемещения и открытия файлов.

```
FileInfo  fi = new FileInfo(string fileName);   //один файл
FileInfo[]  fi = di.GetFiles();     //все файлы директории
```

Одномерные массивы

Массив – это структура данных, хранящая набор значений, идентифицируемых по индексу или набору индексов, принимающих целые значения из некоторого заданного непрерывного диапазона.

Одномерный массив можно рассматривать как реализацию абстрактного типа данных вектор.

Массив в C# является наследником абстрактного класса System.Array из набора System.Collections

Одномерные массивы

Примеры:

```
int[]  arr1 = new  int[2];   //объявляем массив
arr1[0] = 34;    //инициализируем
arr1[1] = 876;

string[]  arr2 = {“art”, “try”, “ball”};

int  el1 = arr1[0];
string  el2 =  arr2[2];
```

Одномерные массивы

Элементы числового массива при создании инициализируются **нулями**

Все лементы строкового массива при создании инициализируются – **null**

Элементы типа bool при создании инициализируются – **false**

Одномерные массивы

BinarySearch()	Этот статический метод можно использовать только тогда, когда массив реализует интерфейс IComparer Если этот интерфейс реализован, метод BinarySearch() позволяет найти элемент массива
Clear()	Этот статический метод позволяет очистить указанный диапазон элементов (числовые элементы приобретут значения 0, а ссылки на объекты — null)
CopyTo()	Используется для копирования элементов из исходного массива в массив назначения
GetLength() Length	Метод GetLength() используется для определения количества элементов в указанном измерении массива. Length — это свойство только для чтения, с помощью которого можно получить количество элементов массива
GetLowerBound() GetUpperBound()	Эти методы используются для определения нижней и верхней границы выбранного вами измерения массива
GetValue() SetValue()	Возвращает или устанавливает значение указанного индекса для массива. Этот метод перегружен для нормальной работы как с одномерными, так и с многомерными массивами
Reverse()	Этот статический метод позволяет расставить элементы одномерного массива в обратном порядке
Sort()	Сортирует одномерный массив встроенных типов данных. Если элементы массива поддерживают интерфейс IComparer, то с помощью этого метода вы сможете производить сортировку и ваших пользовательских типов данных

Динамические массивы

Объявление массива:

```
ArrayList  arrLst1 = new ArrayList();
```

Добавляем эл-ты:

```
arrLst1.Add(«Иванов Иван»);
arrLst1.Add(45);
```

Перебираем эл-ты массива:

```
for (int i = 0; i < arrLst1.Count; i++)
      MessageBox.Show(arrLst1[i].ToString());
string  element1  = (string)arrLst1[0];
int  element 2 = (int)arrLst1[0];   // выдаст ошибку
```

Многомерные массивы

Примеры:

Создание двумерного массива из 10x20 элементов
типа int:

```
int [,]  intArray = new int [10, 20];
```

Создание 3-мерного массива из 12x10x5 элементов типа string:

```
string [,,] stringArray = new string [12, 10, 5];
```

Многомерные массивы

Инициализируем массив 3x2 при создании:

```
int [,] arr1 = new  int[,] { {0, 3}, {7, 17}, {25, 0} };
```

укороченный вариант:

```
int [,] arr2 = { {0, 3}, {7, 17}, {25, 0} };
```

Многомерные массивы

Можно объявить массив без инициализации:

```
int [,] arr3;
```

Нельзя использовать, пока он не создан с помощью оператора new!

```
arr3[0,0] = 0;      // ошибка
arr3 = { {0, 3}, {7, 17}, {25, 0} };    // ошибка

arr3 = new int[,] { {0, 3}, {7, 17}, {25, 0} };     // верно

arr3[0, 0] = 10;    //присваиваем новые значения
```

Многомерные массивы

Создание массива 2 x 3 x 4:

```
int[, ,] arr2 = new int[, ,]
{
    { { 0, 3,2,5 }, { 7, 17,56,47 }, { 25, 0,1,24 } },
    { { 12, 45,18, 20 }, { 172, 11,34,10 }, { 233, 15,61,42 } }
};
```

Что выйдет в тексте следующего диалогового окна?

```
MessageBox.Show(arr2[1, 1, 0].ToString());
```

Многомерные массивы

Задача:

Дан одномерный массив размерностью N.
Необходимо заполнить его случайными числами в диапазоне от -500 до 500.
Подсчитать количество отрицательных чисел.

Многомерные массивы

Решение:

```
label1.Text = "";
int[] arr1 = new int[5];

Random intRnd = new Random();  //класс для случайных величин
int  count = 0;    //для подсчета отрицательных чисел
for (int i = 0; i < 5; i++)
{
   arr1[i] = intRnd.Next(-500, 501); //число от -500 до 500
   if (arr1[i] < 0)  count++;   //увеличиваем на единицу

   label1.Text += arr1[i] + " ";  //составляем строку
}
label2.Text = "Количество отрицательных чисел = "+
count.ToString();
```

Зубчатые (рваные) массивы

Зубчатый массив хранит в качестве элементов другие массивы, и каждый из этих массивов может иметь произвольную размерность.

Объявление массива:

```
int [][] intArray = new int [3][];
```

Инициализация массива:

```
intArray[0] = new int[] {1, 2, 3, 4, 5};
intArray[1] = new int[] {3, 4, 5};
intArray[2] = new int[] {1, 2, 3, 4, 5, 6, 7};
```

Зубчатые (рваные) массивы

Варианты инициализации:

```
int[][] myArray = new int [][]
{
   new int[] {1, 2, 3, 4, 5},
   new int[] {3, 4, 5},
   new int[] {1, 2, 3, 4, 5, 6, 7}
};

int[][] myArray =
{
   new int[] {1, 2, 3, 4, 5},
   new int[] {3, 4, 5},
   new int[] {1, 2, 3, 4, 5, 6, 7}
};
```

Зубчатые (рваные) массивы

Что выйдет в диалоговом окне?

```
int[][] myArray = new int[][]
{
      new int[] {1, 2, 3, 4, 5},
      new int[] {23, 24, 25},
      new int[] {31, 32, 33, 34, 35, 36, 37}
};

MessageBox.Show(myArray[1][2].ToString());
```

Зубчатые (рваные) массивы

Зубчатые и обычные массивы можно смешивать:

```
int [][,] myArray = new int [3][,]
{
   new int[,] { {1, 2}, {3, 4} },
   new int[,] { {5, 6}, {7, 8}, {9, 10} },
   new int[,] { {11, 12} }
};
```

Доступ к элементу массива:

```
MessageBox.Show(myArray[0][0,0].ToString());
```

Зубчатые (рваные) массивы

Можно объявить массив без инициализации:

```
int [][] arr3;
```

Но его нельзя использовать, пока он не создан с помощью оператора new!

Зубчатые массивы используются тогда, когда заранее не известны размеры массивов, выступающих в качестве элементов массива.

Зубчатые (рваные) массивы

Задача:

Сформировать строку, состоящую из произвольного количества элементов массива латинских символов. Массив может быть произвольного размера.

Зубчатые (рваные) массивы

Решение:

```
label1.Text = "";
label2.Text = "";

//создаем вспомогательный (рабочий) массив:
char[] alph = new char[26];

//заполняем рабочий массив латинскими символами:
int  j = 0;
for (char  i = 'A'; i < 'Z'; i++) //цикл по лат. алфавиту
{
    alph[j] = i;
    j++;
}
```

Зубчатые (рваные) массивы

```
Random  rnd = new Random(); //класс случайных значений
int t = rnd.Next(1, 10);     //задаем размер массива

char[] arrSymb = new char[t];   //создаем массив

//заполняем массив символами из рабочего массива
for (int i = 0; i < t; i++)
{
   arrSymb[i] = alph[rnd.Next(0, 26)];
   label1.Text += arrSymb[i];
}
```

Лабораторная работа 3

1. Сформировать массив строк, где каждая строка имеет произвольный размер. Записать сформированный массив строк в текстовый файл.
   В задании использовать зубчатый массив, строки формировать с помощью класса Random.

2. Используя текстовый файл из предыдущего задания, подсчитать количество символа ‘S’ для каждой строки.

StreamReader sr = new StreamReader(“C:\\...\\<fileName>.txt”);
while (sr.Peek() != -1) { } – цикл до конца файла
string str = sr.ReadLine() – считываем всю строку
str.Length – длина строки
str[i] – обращение к i-му символу в строке