**ПРАКТИЧЕСКАЯ РАБОТА № 6. Программирование с использованием одномерных массивов**

***Цель практической работы****:* Изучить способы получения случайных чисел. Написать программу для работы с одномерными массивами.

***6.1. Работа с массивами***

Массив - набор элементов одного и того же типа, объединенных общим именем. Массивы в С# можно использовать по аналогии с тем, как они используются в других языках программирования. Однако С#-массивы имеют существенные отличия: они относятся к ссылочным типам данных, более того - реализованы как объекты. Фактически имя массива является ссылкой на область кучи (динамической памяти), в которой последовательно размещается набор элементов определенного типа. Выделение памяти под элементы происходит на этапе инициализации массива. А за освобождением памяти следит система сборки мусора - неиспользуемые массивы автоматически утилизируются данной системой.

Рассмотрим в данной практической работе одномерные массивы. *Одномерный массив* - это фиксированное количество элементов одного и того же типа, объединенных общим именем, где каждый элемент имеет свой номер. Нумерация элементов массива в С# начинается с нуля, то есть, если массив состоит из 10 элементов, то его элементы будут иметь следующие номера: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Одномерный массив в С# реализуется как объект, поэтому его создание представляет собой двухступенчатый процесс. Сначала объявляется ссылочная переменная на массив, затем выделяется память под требуемое количество элементов базового типа, и ссылочной переменной присваивается адрес нулевого элемента в массиве. Базовый тип определяет тип данных каждого элемента массива. Количество элементов, которые будут храниться в массиве, определяется размер массива.

В общем случае процесс объявления переменной типа массив, и выделение необходимого объема памяти может быть разделено. Кроме того, на этапе объявления массива можно произвести его инициализацию. Поэтому для объявления одномерного массива может использоваться одна из следующих форм записи:

**базовый\_тип [] имя\_\_массива;**

Описана ссылка на одномерный массив, которая в дальнейшем может быть использована для адресации на уже существующий массив. Например:  **int [] a;**

**базовый\_тип [] имя\_\_массива = new базовый\_тип [размер];**

Объявлен одномерный массив заданного типа и выделена память под одномерный массив указанной размерности. Адрес данной области памяти записан в ссылочную переменную. Элементы массива равны нулю (В C# элементам массива присваиваются начальные значения по умолчанию в зависимости от базового типа. Для арифметических типов - нули, для ссылочных типов - null, для символов - пробел). Например: **int []a=new int [10];**

**базовый\_тип [] имя\_\_массива={список инициализации};**

Выделена память под одномерный массив, размерность которого соответствует количеству элементов в списке инициализации. Адрес этой области памяти записан в ссылочную переменную. Значение элементов массива соответствует списку инициализации. Например: **int []a={0, 1, 2, 3};**

Обращения к элементам массива происходит с помощью индекса, для этого нужно указать имя массива и в квадратных скобках его номер. Например, **a[0]**, **b[10]**, **c[i]**.

Так как массив представляет собой набор элементов, объединенных общим именем, то обработка массива обычно производится в цикле. Например:

**int[] myArray = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };**

**int i;**

**for (i = 0; i < 10; ++i)**

**Console.WriteLine(myArray[i]);**

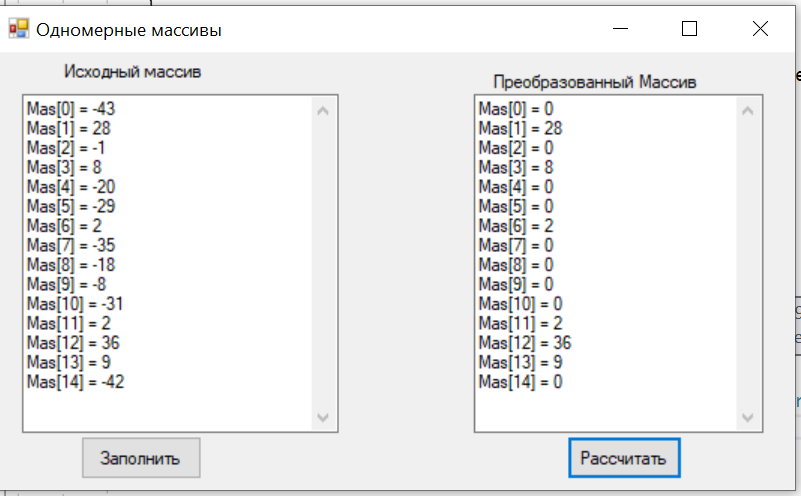
***6.2. Случайные числа***

Одним из способов задания массива является задание определение элементов через случайные числа. Для работы со случайными числами используются в основном два метода класса **Random: Random и Next.** Метод **Random** подготавливает работу со случайными числами, обеспечивая, надежный способ создания непредсказуемой последовательности чисел.

Метод **Random.Next** создает случайное число в диапазоне значений от нуля до **Int32.MaxValue**. Для создания случайного числа в диапазоне от нуля до какого-либо другого положительного числа используется перегрузка метода **Random.Next(Int32)**. Для создания случайного числа в другом диапазоне используется перегрузка метода **Random.Next(Int32, Int32)**.

***6.3. Порядок выполнения индивидуального задания***

Создайте форму с элементами управления как приведено на рис. 6.1. Опишите одномерный массив. Создайте обработчики события для кнопок (код приведен ниже). Данная программа заменяет все отрицательные числа нулями. Протестируйте правильность выполнения программы. Модифицируйте программу в соответствии с индивидуальным заданием.



*Рис. 6.1. Окно программы для работы с одномерными массивами*

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

namespace WindowsFormsApplication1

{

public partial class Form1 : Form

{

int[] Mas = new int[15];

public Form1()

{

InitializeComponent();

}

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

{

Random rand = new Random();

textBox1.Text = "";

for (int i = 0; i < 15; i++)

{

Mas[i] = rand.Next(-50, 50);

textBox1.Text += "Mas[" + Convert.ToString(i) +"] = "

+ Convert.ToString(Mas[i]) + Environment.NewLine;

}

}

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

{

textBox2.Text = "";

for (int i = 0; i < 15; i++)

{

if (Mas[i] < 0) Mas[i] = 0;

textBox2.Text += "Mas[" + Convert.ToString(i) + "] = "

+ Convert.ToString(Mas[i]) + Environment.NewLine;

}

}

}

}

***6.3. Варианты заданий***

1. Найти сумму четных элементов массива.

2. Найти сумму положительных элементов массива.

3. Найти произведение элементов массива с четными индексами.

4. Найти количество нулевых и количество отрицательных эле-

ментов массива.

5. Найти сумму всех элементов массива, которые без остатка

делятся на "6".

6. Найти количество положительных элементов массива, не пре-

вышающих числа "7".

7. Заменить отрицательные элементы нулями.

8. Увеличить все положительные элементы массива на единицу.

9. Найти произведение ненулевых элементов массива.

10. Увеличить элементы массива с четными индексами на "1", а

элементы с нечетными индексами на "2".

11. Найти сумму отрицательных элементов массива а произведе-

ние положительных элементов.

12. Найти среднее арифметическое и среднее геометрическое

значения элементов массива.

13. Указать номера элементов массива, которые без остатка

делятся на "3" и на "5".

14. Указать номера элементов массива, которые больше "2", но

меньше "5".

15. Указать, какой процент от общего количества составляют

положительные элементы массива.

16. Сделать все положительные элементы массива отрицательны-

ми, вывести их индексы.

**Практическая РАБОТА 7. Программирование с использованием МНОГОМЕРНЫХ массивов**

***Цель практической работы****:* изучить свойства компонента **dataGridView**. Написать программу с использованием двухмерных массивов.

***7.1. Двухмерные массивы***

Многомерные массивы имеют более одного измерения. Чаще всего используются двумерные массивы, которые представляют собой таблицы. Каждый элемент массива имеет два индекса, первый определяет номер строки, второй - номер столбца, на пересечении которых находится элемент. Нумерация строк и столбцов начинается с нуля. Объявить двумерный массив можно одним из предложенных способов:

**тип [,] имя\_\_массива;**

**тип [,] имя\_\_массива = new тип [размер1, размер2];**

**тип [,] имя\_\_массива={{элементы 1-ой строки}, … , {элементы n-ой строки}};**

**тип [,] имя\_\_массива= new тип [,]{{элементы 1-ой строки}, … ,{элементы n-ой строки}};**

**строки}};**

Пример кода, использующего многомерные массивы:

// объявление и инициализация двухмерного массива

int[,] array2D = new int[,] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

// Объявление такого массива с указанием размерности (кол-во строки столбцов)

int[,] array2Da = new int[4, 2] { { 1, 2 }, { 3, 4 }, { 5, 6 }, { 7, 8 } };

// Объявление двухмерного массива элементами, которого являются строки

string[,] array2Db = new string[3, 2] { { "one", "two" }, { "three", "four" },

{ "five", "six" } };

// Объявление трехмерного массива

int[, ,] array3D = new int[,,] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

// Объявление трехмерного массива с указанием размерности

int[, ,] array3Da = new int[2, 2, 3] { { { 1, 2, 3 }, { 4, 5, 6 } },

{ { 7, 8, 9 }, { 10, 11, 12 } } };

// Доступ к элементам массива

System.Console.WriteLine(array2D[0, 0]);

System.Console.WriteLine(array2D[0, 1]);

System.Console.WriteLine(array2D[1, 0]);

System.Console.WriteLine(array2D[1, 1]);

System.Console.WriteLine(array2D[3, 0]);

System.Console.WriteLine(array2Db[1, 0]);

System.Console.WriteLine(array3Da[1, 0, 1]);

System.Console.WriteLine(array3D[1, 1, 2]);

// Результаты работы программы (выводятся в консоль):

// 1

// 2

// 3

// 4

// 7

// three

// 8

// 12

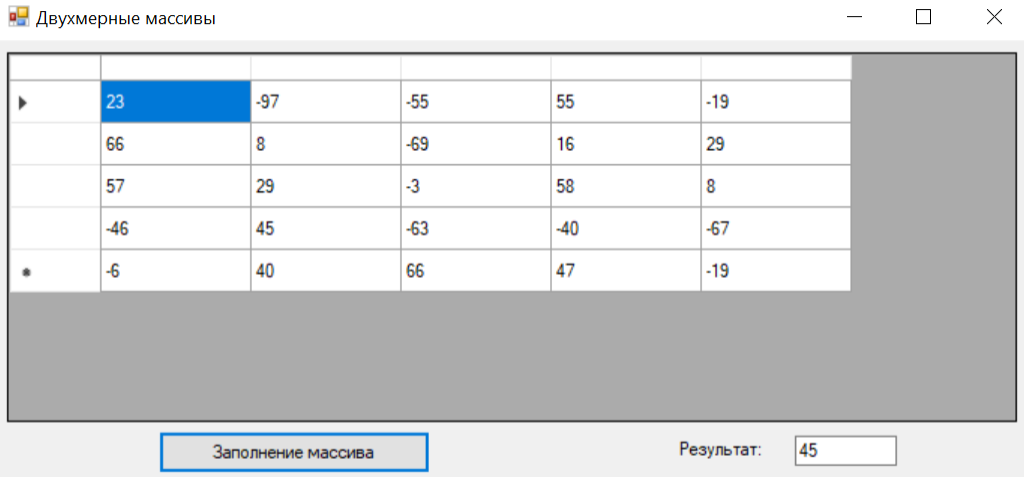
***7.2. Элемент управления DataGridView***

При работе с двухмерными массивами ввод и вывод информации на экран удобно организовывать в виде таблиц. Элемент управления **DataGridView** может быть использован для отображения информации в виде двумерной таблицы. Для обращения к ячейке в этом элементе необходимо указать номер строки и номер столбца. Например: **dataGridView1.Rows[2].Cells[7].Value = "\*";**  данный код позволят записать во вторую строку в 7 ячейку знак звездочка.

***7.3. Порядок выполнения задания***

Задание: Создать программу для определения целочисленной матрицы 5 на 5. Разработать обработчик для поиска минимального элемента на дополнительной диагонали матрицы. Результат, после нажатия кнопки типа **Button**, вывести в **textBox**.

Окно программы приведено на рис. 7.1.



*Рис. 7.1. Окно программы для работы с двухмерным массивом*

Текст обработчика события нажатия на кнопку приведен ниже.

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

{

dataGridView1.RowCount = 15; //Указываем количество строк

dataGridView1.ColumnCount = 15; //Указываем количество столбцов

int[,] a = new int[15,15]; //Инициализируем массив

int i,j;

//Заполняем матрицу случайными числами

Random rand = new Random();

for (i=0; i<15; i++)

for (j=0; j<15; j++)

a[i,j] = rand.Next(-100,100);

//Выводим матрицу в dataGridView1

for (i=0; i<15; i++)

for (j=0; j<15; j++)

dataGridView1.Rows[i].Cells[j].Value = Convert.ToString(a[i,j]);

//производим поиск максимального элемента на дополнительной диагонали

int m = int.MinValue;

for (i = 0; i < 15; i++)

if (a[i, 14 - i] > m) m = a[i, 14 - i];

// выводим результат

textBox1.Text = Convert.ToString(m);

}

***7.4. Индивидуальные задания***

1. Найти максимальный элемент среди элементов находящихся выше главной диагонали квадратной матрицы. Указать его индексы.

2. Найти максимальный и минимальный элементы среди элементов, расположенных на побочной диагонали квадратной матрицы.

3. Поменять местами в матрице минимальный и максимальный элементы.

4. Найти в каждой строке матрицы минимальный элемент. Выделить максимальное из полученных значений.

5. Вычислить, в какой строке матрицы сумма элементов максимальная, а в какой - минимальная.

6. Найти в квадратной матрице сумму элементов, расположенных ниже побочной диагонали.

7. Найти в матрице максимальный и минимальный элементы, поменять местами строки, в которых они расположены.

8. Найти на главной диагонали квадратной матрицы максимальный и минимальный элементы. Поменять местами строки, в которых они расположены.

9. Найти для каждой строки матрицы сумму максимального и минимального элементов.

10. В строке матрицы с максимальным элементом обнулить все элементы, кроме максимального.

11. Для столбцов матрицы с четными номерами найти максимальный элемент, для столбцов с нечетными - минимальный.

12. Сформировать одномерный массив, элементы которого находятся суммированием элементов строк матрицы, в которых находятся максимальный и минимальный элементы.

13. Среди элементов квадратной матрицы, расположенных ниже побочной диагонали найти минимальный и максимальный элементы.

14. Найти в каждой строке матрицы минимальный среди среди положительных элементов.

15. Поменять местами в матрице максимальный и минимальный положительные элементы.

1. Обнулить столбцы матрицы, в которых находятся максимальный и минимальный элементы.