ЛАБОРАТОРНАЯ РАБОТА 4. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ МНОГОМЕРНЫХ МАССИВОВ

Цель лабораторной работы: освоить использование многомерных, динамических и ступенчатых массивов, изучить свойства компонента **dataGridView**.

3.1. Массивы в C#

Массив задает способ организации данных. Массивом называют упорядоченную совокупность элементов одного типа. Каждый элемент массива имеет индексы, определяющие порядок элементов. Число индексов характеризует размерность массива. В языке C#, как и во многих других языках, индексы задаются целочисленным типом. В других языках, например, в языке Паскаль, индексы могут принадлежать счетному конечному множеству, на котором определены функции, задающие следующий и предыдущий элемент.

К сожалению, не снято ограничение 0-базируемости, означающее, что нижняя граница массивов C# фиксирована и равна нулю. Т.е. индексирование элементов всегда начинается с нуля.

3.1.1. Объявление одномерных массивов

Объявление одномерного массива выглядит следующим образом:
<тип>[] <объявители>;

В отличие от языка C++ квадратные скобки приписаны не к имени переменной, а к типу. Они являются неотъемлемой частью определения типа, так что запись T[] следует понимать как тип, задающий одномерный массив с элементами типа T.

Как и в случае объявления простых переменных, каждый объявитель может быть именем или именем с инициализацией. В первом случае речь идет об отложенной инициализации. Нужно понимать, что при объявлении с отложенной инициализацией сам массив не формируется, а создается только ссылка на массив, имеющая неопределенное значение. Поэтому пока массив не будет реально создан и его элементы инициализированы, использовать его в вычислениях нельзя.

Варианты описания массива:

тип[] имя;
тип[] имя = new тип [размерность];
тип[] имя = { список_инициализаторов };
тип[] имя = new тип [] { список_инициализаторов };
тип[] имя = new тип [размерность] { список_инициализаторов };

Примеры описаний:

```
int[] a; // 1 элементов нет
int[] b = new int[4]; // 2 элементы равны 0
int[] c = { 61, 2, 5, -9 }; // 3 new подразумевается
int[] d = new int[] { 61, 2, 5, -9 }; // 4 размерность
вычисляется, избыточное описание
```

```
int[] e = new int[4] { 61, 2, 5, -9 }; // 5 избыточное описание
```

3.1.2. Динамические массивы

Во всех вышеприведенных примерах объявлялись статические массивы, поскольку нижняя граница равна нулю по определению, а верхняя всегда задавалась в этих примерах константой. Статические массивы скорее исключение, а на практике в основном используются динамические массивы.

Чисто синтаксически нет существенной разницы в объявлении статических и динамических массивов. Выражение, задающее границу изменения индексов, в динамическом случае содержит переменные. Единственное требование - значения переменных должны быть определены в момент объявления. Это ограничение в C# выполняется, поскольку C# контролирует инициализацию переменных.

Рассмотрим пример, в котором описана работа с динамическим массивом:

```
static void TestDynAr()  {
//объявление динамического массива A1
      Console.WriteLine("Введите число элементов массива A1");
      int size = int.Parse(Console.ReadLine());
      int[] A1 = new int[size];
      Arrs.CreateOneDimAr(A1);
      Arrs.PrintAr1("A1",A1);
}
```

Здесь верхняя граница массива определяется пользователем.

3.1.3. Многомерные массивы

Очевидно, что одномерные массивы - это частный случай многомерных. Одномерные массивы позволяют задавать такие математические структуры, как векторы, двумерные - матрицы, трехмерные – кубы данных, массивы большей размерности – многомерные кубы данных.

Размерность массива это характеристика типа. Синтаксически размерность массива задается за счет использования запятых. Вот как выглядит объявление многомерного массива в общем случае:

<тип>[, … ,] <объявители>;

Число запятых, увеличенное на единицу, и задает размерность массива. Что касается объявителей, то все, что сказано для одномерных массивов, справедливо и для многомерных. Можно лишь отметить, что хотя явная инициализация с использованием многомерных константных массивов возможна, но применяется редко из-за громоздкости такой структуры. Проще инициализацию реализовать программно, но иногда она все же применяется:

```
 int[,]matrix = {
      {1,2},
      {3,4}
 };
```

Чаще всего в программах используются двумерные массивы. Варианты описания двумерного массива:

тип[,] имя;
тип[,] имя = new тип [разм_1, разм_2];
тип[,] имя = { список_инициализаторов };
тип[,] имя = new тип [,] { список_инициализаторов };
тип[,] имя = new тип [разм_1, разм_2] { список_инициализаторов };

Примеры описаний (один пример на каждый вариант описания):

```
int[,] a; // 1 элементов нет
int[,] b = new int[2, 3]; // 2 элементы равны 0
int[,] c = {{1, 2, 3}, {4, 5, 6}}; // 3 new подразумевается
int[,] c = new int[,] {{1, 2, 3}, {4, 5, 6}}; // 4 размерность вычисляется, избыточно
int[,] d = new int[2,3] {{1, 2, 3}, {4, 5, 6}};// 5 избыточное описание
```

К элементу двумерного массива обращаются, указывая номера строки и столбца, на пересечении которых он расположен, например:

```
a[1, 4];
b[i, j];
b[j, i];
```

3.1.4. Ступенчатые массивы

Еще одним видом массивов C# являются массивы массивов, называемые также ступенчатыми массивами (jagged arrays). Такой массив массивов можно рассматривать как одномерный массив, его элементы являются массивами, элементы которых, в свою очередь снова могут быть массивами, и так может продолжаться до некоторого уровня вложенности. Эти массивы могут применяться для представления деревьев, у которых узлы могут иметь произвольное число потомков. Таковым может быть, например, генеалогическое дерево. Вершины первого уровня - Fathers, представляющие отцов, могут задаваться одномерным массивом, так что Fathers[i] - это i-й отец. Вершины второго уровня представляются массивом массивов - Children, так что Children[i] - это массив детей i-го отца, а Children[i][j] - это j-й ребенок i-го отца. Для представления внуков понадобится третий уровень, так что GrandChildren [i][j][k] будет представлять к-го внука j-го ребенка i-го отца.

Есть некоторые особенности в объявлении и инициализации таких массивов. Если при объявлении типа многомерных массивов для указания размерности использовались запятые, то для изрезанных массивов применяется более ясная символика - совокупности пар квадратных скобок; например, int[][] задает массив, элементы которого - одномерные массивы элементов типа int.

Сложнее с созданием самих массивов и их инициализацией. Здесь нельзя вызвать конструктор new int[3][5], поскольку он не задает изрезанный массив. Фактически нужно вызывать конструктор для каждого массива на самом нижнем уровне. В этом и состоит сложность объявления таких массивов.

```
//массив массивов - формальный пример
 //объявление и инициализация
int[][] jagger = new int[3][] {
        new int[] {5,7,9,11},
        new int[] {2,8},
         new int[] {6,12,4}
};
```

Массив jagger имеет всего два уровня. Можно считать, что у него три элемента, каждый из которых является массивом. Для каждого такого массива необходимо вызвать конструктор new, чтобы создать внутренний массив. В данном примере элементы внутренних массивов получают значение, будучи явно инициализированы константными массивами.

3.1.5. Оператор foreach

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных. Массив является именно такой группой. Удобство этого вида цикла заключается в том, что нам не требуется определять количество элементов в группе и выполнять их перебор по индексу: мы просто указываем на необходимость перебрать все элементы группы. Синтаксис оператора:

foreach (тип имя in выражение) тело_цикла

Имя задает локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из массива выражение (в качестве выражения чаще всего применяется имя массива или другой группы данных). В простом или составном операторе, представляющем собой тело цикла, выполняются действия с переменной цикла. Тип переменной должен соответствовать типу элемента массива. Например, пусть задан массив:

```
int[] a = { 24, 50, 18, 3, 16, -7, 9, -1 };
```

Вывод этого массива на экран с помощью оператора foreach выглядит следующим образом:

```
foreach ( int x in a ) Console.WriteLine( x );
```

Этот оператор выполняется так: на каждом проходе цикла очередной элемент массива присваивается переменной x и с ней производятся действия, записанные в теле цикла.

3.1.6. Методы и свойства класса Array

При обработке массивов могут быть полезными методы и свойства класса Array. Некоторые элементы этого класса показаны в таблице:

Элемент	**Вид**	**Описание**
Length	Свойство	Количество элементов массива (по всем размерностям)
BinarySearch	Статический метод	Двоичный поиск в отсортированном массиве

Clear	Статический метод	Присваивание элементам массива значений по умолчанию
Copy	Статический метод	Копирование заданного диапазона элементов одного массива в другой массив
GetValue	Метод	Получение значения элемента массива
IndexOf	Статический метод	Поиск первого вхождения элемента в одномерный массив
Reverse	Статический метод	Изменение порядка следования элементов на обратный
Sort	Статический метод	Упорядочивание элементов одномерного массива

3.1.7. 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
```

3.2. 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 позволяте записать во вторую строку в 7 ячейку знак звездочка.

3.3. Порядок выполнения задания

Задание: Создать программу для определения целочисленной матрицы 15 на 15. Разработать обработчик для поиска минимального элемента на дополнительной диаганали матрицы. Результат, после нажатия кнопки типа **Button**, вывести в **textBox**.

Окно программы приведено на рис. 3.1.

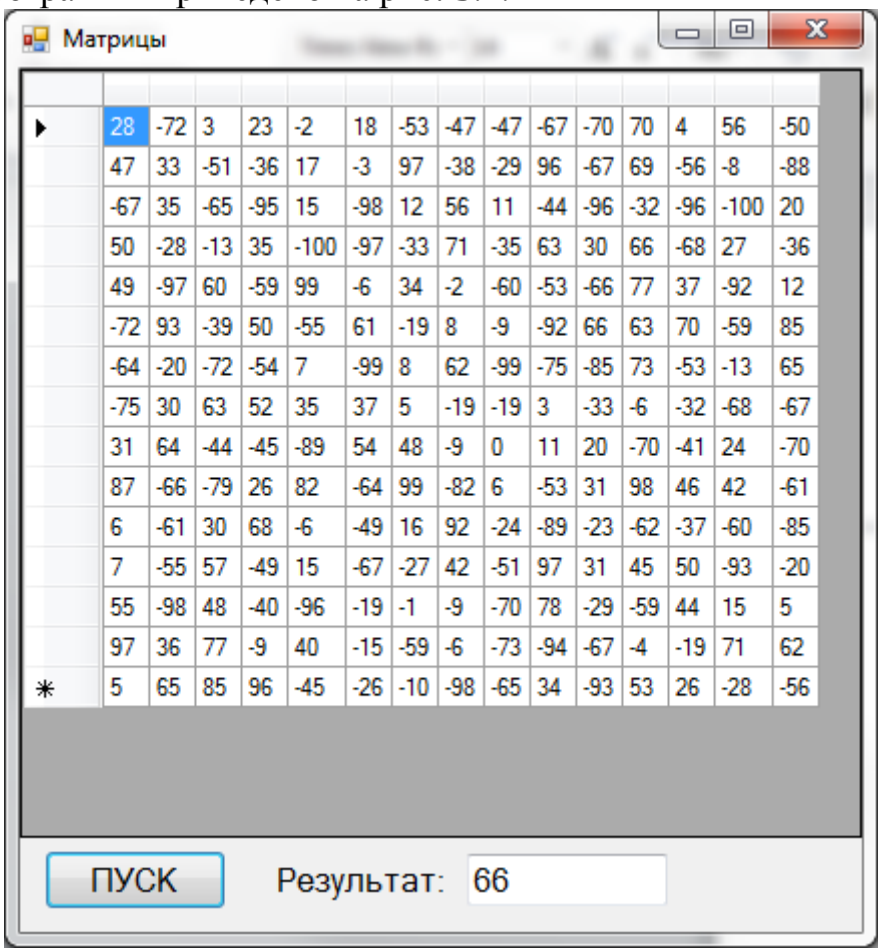

Рис. 3.1. Окно программы для работы с двухмерным массивом

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);
}
```

3.4. Индивидуальные задания

Под словами «дана матрица» («дан массив») следует понимать матрицу (динамический массив), сгенерированную в разрабатываемой программе с использованием генератора случайных чисел. Размерности матриц (массивов), типы данных их элементов (целые/вещественные) и диапазоны значений должны задаваться пользователем при работе программы. Следует максимально использовать возможности, предоставляемые классом Array и оператором foreach. Исходные данные и результаты вычислений должны отображаться в разных элементах управления на форме.

1. Дана прямоугольная целочисленная двумерная матрица. Найти наименьшие элементы в каждой строке матрицы, результат оформить в виде ступенчатого массива, элементы которого для каждой строки содержат индексы найденных элементов.
2. Дана квадратная целочисленная матрица. Построить два ступенчатх массива, содержащие ненулевые элементы, находящиеся выше и ниже побочной диагонали.
3. Дана квадратная вещественная матрица. Найти наибольшие элементы во всех ее диагоналях (включая диагонали длины 1), сформировать ступенчатый массив результатов.
4. Даны две прямоугольные вещественные матрицы. Вычислить их произведение, сформировать ступенчатый массив и занести в него только неотрицательные элементы произведения исходных матриц.
5. Из заданной целочисленной прямоугольной трехмерной матрицы удалить все элементы, значения которых совпадают с суммой индексов. Результат поместить в трехмерный ступенчатый массив.
6. Преобразовать заданную двумерную прямоугольную вещественную матрицу в ступенчатый массив путем удаления из строк тех элементов, значения которых находятся в диапазоне между средним арифметическим данной строки и средним арифметическим всей матрицы.
7. Преобразовать сильно разреженную двумерную целочисленную матрицу (значения большинства элементов которой равны нулю) в два ступенчатых массива. Для каждой строки первый массив содержит значение ненулевого элемента, а второй – индекс столбца исходной матрицы, содержащего этот элемент.

8. Извлечь из каждого столбца заданного двумерного массива в точности n+1 (где n – индекс столбца) наименьших элементов и сформировать ступенчатый треугольный массив, строки которого содержат извлеченные элементы.

9. Преобразовать заданную двумерную матрицу вещественных чисел к верхнему треугольному виду (см. метод Гаусса решения систем линейных алгебраических уравнений), результат оформить в виде ступенчатого массива.

10. Преобразовать заданную целочисленную двумерную матрицу в ступенчатый массив, содержащий в каждой строке только те элементы строки исходной матрицы, которые не делятся нацело на заданное число.

11. Из двух заданных двумерных прямоугольных целочисленных массивов с одинаковым количеством строк построить ступенчатый двумерный массив, строки которого содержат только элементы, имеющиеся в строках обеих исходных массивов с тем же индексом.

12. Разбить заданную прямоугольную двумерную целочисленную матрицу на два ступенчатых массива. В первый массив включить элементы строк до последнего максимального элемента строки исключительно (без этого элемента), во второй массив – все остальные элементы строк.

13. Преобразовать заданную двумерную прямоугольную целочисленную матрицу в ступенчатый массив путем удаления из ее столбцов всех совпадающих элементов, кроме первого.

14. Преобразовать заданную двумерную прямоугольную вещественную матрицу в ступенчатый массив путем удаления из строк тех элементов, которые меньше следующего элемента.

15. Преобразовать заданную двумерную прямоугольную вещественную матрицу в ступенчатый массив путем удаления из столбцов тех элементов, значения которых меньше среднего арифметического, вычисленного для данного столбца.

16. Разбить двумерную прямоугольную вещественную матрицу на два ступенчатых массива. В первый построчно включить те элементы, значения которых меньше среднего арифметического строки, расположив их по возрастанию, во второй – все остальные по убыванию.

17. Преобразовать заданную трехмерную целочисленную матрицу в ступенчатый трехмерный массив, включив в него только уникальные элементы строк исходной матрицы.

18. Даны два одномерных целочисленных массива. Сформировать ступенчатый массив, содержащий в каждой строке арифметическую прогрессию с шагом 1, начинающуюся со значения соответствующего элемента первого массива, длина которой равна значению элемента второго массива (не создавать подмассив, если это значение меньше или равно нулю).

19. Преобразовать сильно разреженную двумерную целочисленную матрицу (значения большинства элементов которой равны нулю) в ступенчатый массив, содержащий пары: количество нулевых элементов – значение ненулевого элемента.

20. На основании заданного двумерного целочисленного массива сформировать два ступенчатых массива, первый из которых содержит уникальные значения элементов строк, а второй – количество этих элементов в строке.

21. Преобразовать заданный двумерный вещественный массив в ступенчатый путем включения в него элементов, сумма индесов строки и столбца которых кратна модулю разности этих индексов плюс 2.

22. Преобразовать заданный двумерный целочисленный массив в ступенчатый, включив в него только те элементы, значения которых равны сумме значений предыдущего и последующего элементов в строке.

23. Разбить заданный двумерный целочисленный массив на два ступенчатых массива, первый из которых содержит все элементы с четными значениями, а второй – все элементы с нечетными значениями.

24. Преобразовать заданную двумерную прямоугольную целочисленную матрицу в ступенчатый массив путем удаления из ее строк всех совпадающих элементов.

25. Извлечь из каждой строки заданного двумерного целочисленного массива столько элементов (начиная с первого элемента строки), каково значение элемента на главной диагонали и сформировать ступенчатый массив.