Многомерные 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}};
```

Например:

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

Замечания.

1. Как и в случае с одномерными массивами, последние два описания являются избыточными.
2. При работе с многомерными массивами можно использовать приемы, которые мы рассмотрели для одномерных массивов.
3. При обращении к свойству Length для двумерного массива мы получим общее количество элементов в массиве. Чтобы получить количество строк нужно обратиться к методу GetLength с параметром 0. Чтобы получить количество столбцов - к методу GetLengthc параметром 1.

Пример:

```
class Program
{
	static void PrintArray(string a, int[,] mas)
	{
		Console.WriteLine(a);
		for (int i = 0; i < mas.GetLength(0); i++)
		{
			for (int j = 0; j < mas.GetLength(1); j++)
			Console.Write("{0} ", mas[i, j]);
			Console.WriteLine();
		}
	}
	static void Change(int[,] mas)
	{
		for (int i = 0; i < mas.GetLength(0); i++)
		for (int j = 0; j < mas.GetLength(1); j++)
		if (mas[i, j] % 2 == 0) mas[i, j] = 0;
	}
	static void Main()
	{
		try
		{
			int[,] MyArray ={ { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
			PrintArray("исходный массив:", MyArray);
```

```
                Change(MyArray);
                PrintArray("итоговый массив", MyArray);
            }
            catch (FormatException)
            {
                Console.WriteLine("неверный формат ввода данных");
            }
            catch (OverflowException)
            {
                Console.WriteLine("переполнение");
            }
            catch (OutOfMemoryException)
            {
                Console.WriteLine("недостаточно памяти для создания нового объекта");
            }
        }
}
```

Ступенчатые массивы

В ступенчатых массивах количество элементов в разных строках может быть различным. В памяти ступенчатый массив хранится в виде массива массивов. Структура ступенчатого массива:

Массив	a	a[0]	a[0][0]	a[0][1]	…
		a[1]			
		…	a[1][0]	a[1][1]	…
		a[n]			
			a[n][0]	a[n][1]	…

Объявление ступенчатого массива:

тип [][] имя_массива;

Например:

int [][]a;

Фактически мы объявили одномерный массив ссылок на целочисленные одномерные массивы. При таком описании потребуется не только выделять память под одномерный массив ссылок, но и под каждый из целочисленных одномерных массивов. Такое распределение памяти позволяет определять произвольную длину каждой строки массива (отсюда и произошло название массива - ступенчатый).

```
int [][] a= new int [3][];  // Создаем три строки
a[0]=new int [2];  // 0-ая строка ссылается на 2-х элементый одномерный массив
a[1]=new int [3];  // 1-ая строка ссылается на 3-х элементый одномерный массив
a[2]=new int [10];  // 2-ая строка ссылается на 10-ти элементый одномерный массив
```

Другой способ выделения памяти:

```
int [][] a= {new int [2], new int [3], new int [10]};
```

Так как каждая строка ступенчатого массива фактически является одномерным массивом, то с каждой строкой можно работать как с экземпляром класса Array. Это является преимуществом ступенчатых массивов перед двумерными массивами.

Пример:

```
class Program
{
        static void Main()
        {
                try
                {
                        int[][] MyArray;
                        Console.Write("Ввведите количество строк: ");
                        int n = int.Parse(Console.ReadLine());
                        MyArray = new int[n][];
                        for (int i = 0; i < MyArray.Length; i++)
                        {
                                Console.Write("введите количество элементов в {0} строке: ", i);
                                int j = int.Parse(Console.ReadLine());
                                MyArray[i] = new int[j];
                                for (j = 0; j < MyArray[i].Length; j++)
                                {
                                        Console.Write("a[{0}][{1}]= ", i, j);
                                        MyArray[i][j] = int.Parse(Console.ReadLine());
                                }
                        }
                        PrintArray("исходный массив:", MyArray);
                        for (int i = 0; i < MyArray.Length; i++) Array.Sort(MyArray[i]);
                        PrintArray("измененный массив", MyArray);
```

```
            }
            catch (FormatException)
            {
                Console.WriteLine("неверный формат ввода данных");
            }
            catch (OverflowException)
            {
                Console.WriteLine("переполнение");
            }
            catch (OutOfMemoryException)
            {
                Console.WriteLine("недостаточно памяти для создания нового объекта");
            }
        }
        static void PrintArray(string a, int[][] mas)
        {
            Console.WriteLine(a);
            for (int i = 0; i < mas.Length; i++)
            {
                for (int j = 0; j < mas[i].Length; j++) Console.Write("{0} ", mas[i][j]);
                Console.WriteLine();
            }
        }
}
```

Оператор foreach и его использование при работе с массивами

Оператор foreach применяется для перебора элементов в специальным образом организованной группе данных, в том числе и в массиве. Синтаксис оператора:

foreach (<тип> <имя> in <группа>) <тело цикла>

где *имя* определяет локальную по отношению к циклу переменную, которая будет по очереди принимать все значения из указанной *группы*, а *тип* соответствует базовому типу элементов *группы*.

Ограничением оператора foreach является то, что с его помощью можно только просматривать значения элементов в группе данных, но нельзя их изменять.

Рассмотрим несколько примеров использования оператора foreach:

1. для работы с одномерными массивами:

```
static void PrintArray(string a, int [] mas)
{
        Console.WriteLine(a);
        foreach (int x in mas)Console.Write("{0} ", x);
        Console.WriteLine();
}
```

2. для работы с двумерными массивами:

```
static int Sum (int [,] mas)
{
        int s=0;
        foreach (int x in mas) s += x;
        return s;
}
```

3. для работы со ступенчатыми массивами:

```
static void PrintArray3(string a, int[][] mas)
{
        Console.WriteLine(a);
        for (int i = 0; i < mas.Length; i++)
        {
                foreach (int x in mas[i]) Console.Write("{0} ", x);
                Console.WriteLine();
        }
}
```

Пример. Дана последовательность целых чисел. Заменить все положительные элементы противоположными им числами.

Пример 1: для одномерного массива

```
using System;
namespace ConsoleApplication2
{
	class Class
	{
		static int [] Input ()
		{
			Console.WriteLine("введите размерность массива");
			int n=int.Parse(Console.ReadLine());
			int []a=new int[n];
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}
		static void Print(int[] a)
		{
			for (int i = 0; i < a.Length; ++i) Console.Write("{0} ", a[i]);
			Console.WriteLine();
		}
```

```
        static void Change(int[] a)
        {
            for (int i = 0; i < a.Length; ++i)
            if (a[i] > 0) a[i] = -a[i];
        }
        static void Main()
        {
            int[] myArray=Input();
            Console.WriteLine("Исходный массив:");
            Print(myArray);
            Change(myArray);
            Console.WriteLine("Измененный массив:");
            Print(myArray);
        }
    }
}
```

Пример 2: для двумерного массива

```
using System;
namespace ConsoleApplication
{
        class Class
        {
                static int [,] Input (out int n, out int m)
                {
                        Console.WriteLine("введите размерность массива");
                        Console.Write("n = ");
                        n=int.Parse(Console.ReadLine());
                        Console.Write("m = ");
                        m=int.Parse(Console.ReadLine());
                        int [,]a=new int[n, m];
                        for (int i = 0; i < n; ++i)
                        for (int j = 0; j < m; ++j)
                        {
                           Console.Write("a[{0},{1}]= ", i, j);
                           a[i, j]=int.Parse(Console.ReadLine());
                        }
                        return a;
                }
```

```
        static void Print(int[,] a)
        {
            for (int i = 0; i < a.GetLength(0); ++i,Console.WriteLine() )
            for (int j = 0; j < a.GetLength(1); ++j)
            Console.Write("{0,5} ", a[i, j]);
        }

        static void Change(int[,] a)
        {
            for (int i = 0; i < a.GetLength(0); ++i)
            for (int j = 0; j < a.GetLength(1); ++j)
            if (a[i, j] > 0) a[i, j] = -a[i, j];
        }

        static void Main()
        {
            int n,m;
            int[,] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray);
            Change(myArray);
            Console.WriteLine("Измененный массив:");
            Print(myArray);
        }
    }
}
```

Пример. Дана последовательность из n действительных чисел. Подсчитать количество максимальных элементов.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [] Input ()
		{
			Console.WriteLine("введите размерность массива");
			int n=int.Parse(Console.ReadLine());
			int []a=new int[n];
			for (int i = 0; i < n; ++i)
			{
			Console.Write("a[{0}]= ", i);
			a[i]=int.Parse(Console.ReadLine());
		}
		return a;
	}

	static int Max(int[] a)
	{
		int max=a[0];
		for (int i = 1; i < a.Length; ++i)
		if (a[i] > max) max=a[i];
```

```
            return max;
        }

        static void Main()
        {
            int[] myArray=Input();
            int max=Max(myArray);
            int kol=0;
            for (int i=0; i<myArray.Length;++i)
            if (myArray[i]==max)++kol;
            Console.WriteLine("Количество максимальных элементов = "+kol);
        }
        }
}
```

Пример. Дан массив размером nxn, элементы которого целые числа. Подсчитать среднее арифметическое нечетных элементов, расположенных выше главной диагонали.

```
using System;
namespace ConsoleApplication
{
        class Class
        {
                static int [,] Input (out int n)
                {
                        Console.WriteLine("введите размерность массива");
                        Console.Write("n = ");
                        n=int.Parse(Console.ReadLine());
                        int [,]a=new int[n, n];
                        for (int i = 0; i < n; ++i)
                        for (int j = 0; j < n; ++j)
                        {
                                Console.Write("a[{0},{1}]= ", i, j);
                                a[i, j]=int.Parse(Console.ReadLine());
                        }
                        return a;
                }
```

```
static void Print(int[,] a)
{
        for (int i = 0; i < a.GetLength(0); ++i,Console.WriteLine() )
        for (int j = 0; j < a.GetLength(1); ++j)
        Console.Write("{0,5} ", a[i, j]);
}

static double Rezalt(int[,] a)
{
        int k=0;
        double s=0;
        for (int i = 0; i < a.GetLength(0); ++i)
        for (int j = i+1; j < a.GetLength(1); ++j)
        if (a[i, j] %2!= 0) {++k; s+=a[i, j];}
        if (k!=0) return s/k;
        else return 0;
}

static void Main()
{
        int n;
        int[,] myArray=Input(out n);
        Console.WriteLine("Исходный массив:");
        Print(myArray);
        double rez=Rezalt(myArray);
        Console.WriteLine("Среднее арифметическое ={0:f2}", rez);
```

```
                }
            }
        }
```

Пример. Дан массив размером nxn, элементы которого целые числа. Найти максимальный элемент в каждой строке и записать данные в новый массив.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [][] Input ()
		{
			Console.WriteLine("введите размерность массива");
			Console.Write("n = ");
			int n=int.Parse(Console.ReadLine());
			int [][]a=new int[n][];
			for (int i = 0; i < n; ++i)
			{
				a[i]=new int [n];
				for (int j = 0; j < n; ++j)
				{
					Console.Write("a[{0},{1}]= ", i, j);
					a[i][j]=int.Parse(Console.ReadLine());
				}
			}
			return a;
		}
```

```
static void Print1(int[] a)
{
        for (int i = 0; i < a.Length; ++i)
        Console.Write("{0,5} ", a[i]);
}

static void Print2(int[][] a)
{
        for (int i = 0; i < a.Length; ++i,Console.WriteLine() )
        for (int j = 0; j < a[i].Length; ++j)
        Console.Write("{0,5} ", a[i][j]);
}

static int Max(int[] a)
{
        int max=a[0];
        for (int i = 1; i < a.Length; ++i)
        if (a[i] >max) {max=a[i];}
        return max;
}
```

```
        static void Main()
        {
            int[][] myArray=Input();
            Console.WriteLine("Исходный массив:");
            Print2(myArray);
            int[]rez=new int [myArray.Length];
            for (int i=0;i<myArray.Length; ++i)
            rez[i]=Max(myArray[i]);
            Console.WriteLine("Новый массив:");
            Print1(rez);
        }
    }
}
```

Вставка и удаление элементов в массивах

При объявлении массива мы определяем его максимальную размерность, которая в дальнейшем изменена быть не может. Однако с помощью вспомогательной переменной можно контролировать текущее количество элементов, которое не может быть больше максимального.

Замечание. В пространстве имен System.Collection реализована коллекция *ArrayList* - массив, динамически изменяющий свой размер. Мы будем рассматривать его позже.

Пример. Рассмотрим фрагмент программы:

```
int []a=new int [10];
int n=5;
for (int i=0; i<5;i++) a[i]:=i*i;
```

В этом случае массив можно представить следующим образом:

n=5*01234 56789*

a 01491600000

Так как во время описания был определен массив из 10 элементов, а заполнено только первые 5, то оставшиеся элементы будут заполнены нулями.

Что значит *удалить из одномерного массива* элемент с номером 3? Удаление должно привести к физическому "уничтожению" элемента с номером 3 из массива, при этом общее количество элементов должно быть уменьшено. В этом понимании удаления элемента итоговый массив должен выглядеть следующем образом

0124 56789недопустимое состояние

a0141600000

Такое удаление для массивов *невозможно*, поскольку элементы массива располагаются в памяти последовательно друг за другом, что позволяет организовать индексный способ обращения к массиву.

Однако "удаление" можно смоделировать сдвигом элементов влево и уменьшением значения переменной, которая отвечает за текущее количество элементов в массиве, на единицу:

n=4*0123 456789*

a 01416000000

В общем случае, если мы хотим удалить элемент массива с номером k (всего в массиве n элементов, а последний элемент имеет индекс n-1), то нам необходимо произвести сдвиг элементов, начиная с k+1 -го на одну позицию влево. Т.е. на k -ое место поставить k+1 -й элемент, на место k+1 - k+2 -й элемент, …, на место n-2 - n-1 -й элемент. После чего значение n уменьшить на 1. В этом случае размерность массива не изменится, изменится лишь текущее количество элементов, и у нас создастся ощущение, что элемент с номером k удален.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [] Input ()
		{
			Console.WriteLine("введите размерность массива");
			int n=int.Parse(Console.ReadLine());
			int []a=new int[n];
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}

		static void Print(int[] a, int n)
		{
			for (int i = 0; i < n; ++i) Console.Write("{0} ", a[i]);
			Console.WriteLine();
		}
```

```
        static void DeleteArray(int[] a, ref int n, int m)
        {
            for (int i = m; i < n-1; ++i)
            a[i] = a[i+1];
            --n;
        }

        static void Main()
        {
            int[] myArray=Input();
            int n=myArray.Length;
            Console.WriteLine("Исходный массив:");
            Print(myArray, n);
            Console.WriteLine("Введите номер элемента для удаления:");
            int m=int.Parse(Console.ReadLine());
            DeleteArray(myArray, ref n,m);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n);
        }
    }
}
```

Рассмотрим теперь операцию *удаления в двумерном массиве*. Размерность двумерного массива также зафиксирована на этапе объявления массива. Однако при необходимости можно "смоделировать" удаление целой строки в массиве, выполняя сдвиг всех строк, начиная с k -той на единицу вверх. В этом случае размерность массива не изменится, а текущее количество строк будет уменьшено на единицу. В качестве примера удалим из двумерного массива, строку с номером k.

```
using System;
namespace ConsoleApplication
{
        class Class
        {
                static int [,] Input (out int n, out int m)
                {
                        Console.WriteLine("введите размерность массива");
                        Console.Write("n = ");
                        n=int.Parse(Console.ReadLine());
                        Console.Write("m = ");
                        m=int.Parse(Console.ReadLine());
                        int [,]a=new int[n, m];
                        for (int i = 0; i < n; ++i)
                        for (int j = 0; j < m; ++j)
                        {
                                Console.Write("a[{0},{1}]= ", i, j);
                                a[i, j]=int.Parse(Console.ReadLine());
                        }
                        return a;
```

```
}

static void Print(int[,] a, int n, int m)
{
        for (int i = 0; i < n; ++i,Console.WriteLine() )
        for (int j = 0; j < m; ++j)
        Console.Write("{0,5} ", a[i, j]);
}

static void DeleteArray(int[,] a, ref int n, int m, int k)
{
        for (int i = k; i < n-1; ++i)
        for (int j = 0; j < m; ++j)
        a[i, j] = a[i+1, j];
        --n;
}

static void Main()
{
        int n,m;
        int[,] myArray=Input(out n, out m);
        Console.WriteLine("Исходный массив:");
        Print(myArray, n, m);
        Console.WriteLine("Введите номер строки для удаления:");
        int k=int.Parse(Console.ReadLine());
        DeleteArray(myArray, ref n, m, k);
```

```
                Console.WriteLine("Измененный массив:");
                Print(myArray, n, m);
            }
        }
    }
```

Рассмотрим модификацию предыдущей программы, для случая, когда используется ступенчатый массив.

```
using System;
namespace ConsoleApplication
{
    class Class
    {
        static int [][] Input (out int n, out int m)
        {
            Console.WriteLine("введите размерность массива");
            Console.Write("n = ");
            n=int.Parse(Console.ReadLine());
            Console.Write("m = ");
            m=int.Parse(Console.ReadLine());
            int [] []a=new int[n][];
            for (int i = 0; i < n; ++i)
            {
                a[i]=new int[m];
```

```
                for (int j = 0; j < m; ++j)
                {
                        Console.Write("a[{0},{1}]= ", i, j);
                        a[i][j]=int.Parse(Console.ReadLine());
                }
        }
        return a;
}

static void Print(int[][] a, int n, int m)
{
        for (int i = 0; i < n; ++i, Console.WriteLine() )
        for (int j = 0; j < m; ++j)
        Console.Write("{0,5} ", a[i] [j]);
}

static void DeleteArray(int[][] a, ref int n, int k)
{
        for (int i = k; i < n-1; ++i)//производим сдвиг ссылок
        a[i] = a[i+1];
        --n;
}
```

```
        static void Main()
        {
            int n,m;
            int[][] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для удаления:");
            int k=int.Parse(Console.ReadLine());
            DeleteArray(myArray, ref n, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```

Вернемся к массиву, определенному в самом первом примере. И подумаем теперь, что значит *добавить элемент в одномерный массив* в позицию с номером k? В этом случае все элементы, начиная с k -ого, должны быть сдвинуты вправо на одну позицию. Однако сдвиг нужно начинать с конца, т.е. на первом шаге на n -е место поставить n-1 -ый элемент, потом на n-1 -ое место поставить n-2 -й элемент, …, наконец, на k+ 1 место вставить k-й элемент. Таким образом, копия k -го элемента будет на k+1 -м месте и на k -е место можно поставить новый элемент. Затем необходимо увеличить текущее количество элементов на 1.

Рассмотрим массив из примера 1 и в качестве k зададим значение равное 3. В этом случае массив будет выглядеть следующим образом:

k=3*012345 6789*
a 014**9**9160000

Теперь в позицию с номером 3 можно поместить новое значение. А текущее количество элементов в массиве становится равным 6. Подумайте, почему сдвиг нужно выполнять с конца массива, а не с начала, как мы это делали в случае удаления элемента из массива.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [] Input (out int n)
		{
			Console.WriteLine("введите размерность массива");
			n=int.Parse(Console.ReadLine());
			int []a=new int[2*n]; //выделяем памяти больше чем требуется
			for (int i = 0; i < n; ++i)
			{
				Console.Write("a[{0}]= ", i);
				a[i]=int.Parse(Console.ReadLine());
			}
			return a;
		}
		static void Print(int[] a, int n)
		{
			for (int i = 0; i < n; ++i) Console.Write("{0} ", a[i]);
			Console.WriteLine();
		}
```

```
        static void AddArray(int[] a, ref int n, int m)
        {
            for (int i = n; i >= m; --i)
            a[i] = a[i-1];
            ++n;
            Console.WriteLine("Введите значение нового элемента");
            a[m]=int.Parse(Console.ReadLine());
        }

        static void Main()
        {
            int n;
            int[] myArray=Input(out n);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n);
            Console.WriteLine("Введите номер элемента для вставки:");
            int m=int.Parse(Console.ReadLine());
            AddArray(myArray, ref n,m);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n);
        }
    }
}
```

Рассмотрим *добавление строки в двумерный массив*. Для этого все строки после строки с номером k передвигаем на 1 строку вниз. Затем увеличиваем количество строк на 1. После этого копия строки с номером k будет находиться в столбце с номером k+1. И, следовательно, k -тый столбец можно заполнить новыми значениями.

```
using System;
namespace ConsoleApplication
{
        class Class
        {
                static int [,] Input (out int n, out int m)
                {
                        Console.WriteLine("введите размерность массива");
                        Console.Write("n = ");
                        n=int.Parse(Console.ReadLine());
                        Console.Write("m = ");
                        m=int.Parse(Console.ReadLine());
                        //выделяем памяти больше чем необходимо
                        int [,]a=new int[2*n, m];
                        for (int i = 0; i < n; ++i)
                        for (int j = 0; j < m; ++j)
                        {
                                Console.Write("a[{0},{1}]= ", i, j);
                                a[i, j]=int.Parse(Console.ReadLine());
                        }
                        return a;
```

```
}

static void Print(int[,] a, int n, int m)
{
        for (int i = 0; i < n; ++i,Console.WriteLine() )
        for (int j = 0; j < m; ++j)
        Console.Write("{0,5} ", a[i, j]);
}

static void AddArray(int[,] a, ref int n, int m, int k)
{
        for (int i = n; i >=k; --i)
        for (int j = 0; j < m; ++j)
        a[i+1, j] = a[i, j];
        ++n;
        Console.WriteLine("Введите элементы новой строки");
        for (int j=0; j<m;++j)
        {
                Console.Write("a[{0},{1}]=", k, j);
                a[k, j]=int.Parse(Console.ReadLine());
        }
}
```

```
        static void Main()
        {
            int n,m;
            int[,] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для добавления:");
            int k=int.Parse(Console.ReadLine());
            AddArray(myArray, ref n, m, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```

Рассмотрим модификацию предыдущей программы для случая, когда используется ступенчатый массив.

```
using System;
namespace ConsoleApplication
{
	class Class
	{
		static int [][] Input (out int n, out int m)
		{
			Console.WriteLine("введите размерность массива");
			Console.Write("n = ");
			n=int.Parse(Console.ReadLine());
			Console.Write("m = ");
			m=int.Parse(Console.ReadLine());
			//выделяем памяти больше чем неообходимо
			int [][]a=new int[2*n][];
			for (int i = 0; i < n; ++i)
			{
				a[i]=new int [m];
				for (int j = 0; j < m; ++j)
				{
					Console.Write("a[{0}][{1}]= ", i, j);
					a[i][j]=int.Parse(Console.ReadLine());
				}
			}
```

```
        return a;
}

static void Print(int[][] a, int n, int m)
{
        for (int i = 0; i < n; ++i,Console.WriteLine() )
        for (int j = 0; j < m; ++j)
        Console.Write("{0,5} ", a[i][j]);
}

static void AddArray(int[][] a, ref int n, int m, int k)
{
        for (int i = n; i >=k; --i)//выполняем сдвиг ссылок
        a[i+1] = a[i];
        ++n;
        a[k]=new int[m]; //создаем новую строку
        Console.WriteLine("Введите элементы новой строки");
        for (int j=0; j<m;++j)
        {
                Console.Write("a[{0}][{1}]=", k, j);
                a[k][j]=int.Parse(Console.ReadLine());
        }
}
```

```
        static void Main()
        {
            int n,m;
            int[][] myArray=Input(out n, out m);
            Console.WriteLine("Исходный массив:");
            Print(myArray, n, m);
            Console.WriteLine("Введите номер строки для добавления:");
            int k=int.Parse(Console.ReadLine());
            AddArray(myArray, ref n, m, k);
            Console.WriteLine("Измененный массив:");
            Print(myArray, n, m);
        }
    }
}
```