**Практическое занятие № 2**

**РАБОТА С МАССИВАМИ**

**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. Каждый индекс изменяется в некотором диапазоне [a,b]. В языке C#, как и во многих других языках, индексы задаются целочисленным типом. Диапазон [a,b] называется граничной парой, a – нижней границей, b – верхней границей индекса. При объявлении массива границы задаются выражениями. Если все границы заданы константными выражениями, то число элементов массива известно в момент его объявления и ему может быть выделена память еще на этапе трансляции. Такие массивы называются статическими. Если же выражения, задающие границы, зависят от переменных, то такие массивы называются динамическими, поскольку память им может быть отведена только динамически в процессе выполнения программы, когда становятся известными значения соответствующих переменных. Массиву выделяется непрерывная область памяти. В C# массивы - динамические.

При написании программ, можно создавать одномерные, многомерные массивы и массивы массивов.

***Одномерные массивы***

Объявление одномерного массива выглядит следующим образом:

<тип>[] <имя массива>;

Заметьте, квадратные скобки приписаны не к имени переменной, а к типу. Они являются неотъемлемой частью определения класса, так что, например, запись int [ ] следует понимать как класс одномерный массив с элементами типа int.

Что же касается границ изменения индексов, то эта характеристика к классу не относится, она является характеристикой переменных.

Как и в случае объявления простых переменных, при объявлении массива одновременно может быть проведена и инициализация. Нужно понимать, что при объявлении с отложенной инициализацией сам массив не формируется, а создается только ссылка на массив, имеющая неопределенное значение Null. Поэтому пока массив не будет реально создан и его элементы инициализированы, использовать его в вычислениях нельзя. Пример объявления трех массивов с отложенной инициализацией:

int[] a, b, c;

Чаще всего при объявлении массива используется имя с инициализацией. И опять-таки, как и в случае простых переменных, могут быть два варианта инициализации. В первом случае инициализация является явной и задается константным массивом:

double[] x= {5.5, 6.6, 7.7};

Следуя синтаксису, элементы константного массива следует заключать в фигурные скобки.

Во втором случае создание и инициализация массива выполняется в объектном стиле с вызовом конструктора массива. И это наиболее распространенная практика объявления массивов:

int[] d= new int[5];

Если массив объявляется без инициализации, то создается только ссылка на него. Если инициализация выполняется конструктором, то в динамической памяти создается сам массив, элементы которого инициализируются константами соответствующего типа (ноль для арифметики, пустая строка для строковых массивов), и ссылка связывается с этим массивом. Если массив инициализируется константным массивом, то в памяти создается константный массив, с которым и связывается ссылка.

Рассмотрим еще несколько примеров объявления массивов:

...

int[] k; //k - массив  
k=new int [3]; //Определяем массив из 3-х целых

k[0]=-5; k[1]=4; k[2]=55; //Задаем элементы массива

//Выводим третий элемент массива

Console.WriteLine(k[2].ToString());

...

Смысл приведенного фрагмента ясен из комментариев. Обратите внимание на некоторые особенности. Во-первых, массив определяется именно как

int[] k;

а не как один из следующих вариантов:

Во-вторых, так как массив представляет ссылочный объект, то для создания массива необходима строка

k=new int [3];

Именно в ней мы и определяем размер массива. Хотя, вообще говоря, возможны конструкции вида

int[] k = new int [3];

Элементы массива можно задавать сразу при объявлении. Вот пример:

int[] k = {-5, 4, 55};

Разумеется, приведенные конструкции применимы не только к типу int и не только к массиву размера 3.

В C# нумерация элементов массива начинается с нуля. Таким образом, в примере начальный элемент массива - это k\*0+, а последний - k\*2+. Элемента k\*3+ нет.

***Динамические массивы***

Во всех вышеприведенных примерах объявлялись статические массивы, поскольку нижняя граница равна нулю по определению, а верхняя всегда задавалась в этих примерах константой. В C# все массивы, независимо от того, каким выражением описывается граница, рассматриваются как динамические. В действительности реальные потребности в размере массива, скорее всего, выясняются в процессе выполнения программы.

Синтаксически нет 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.

Пример, в котором описана работа с динамическим массивом:

public void TestDynAr()

{

//объявление динамического массива A1 Console.WriteLine("Введите число элементов массива A1"); int size = int.Parse(Console.ReadLine());  
int[] A1 = new int[size];

}

В данной процедуре верхняя граница массива определяется пользователем.

***Многомерные массивы***

Разделение массивов на одномерные и многомерные носит исторический характер. Никакой принципиальной разницы между ними нет. Одномерные массивы -это частный случай многомерных. Можно говорить и по-другому: многомерные массивы являются естественным обобщением одномерных. Одномерные массивы позволяют задавать такие математические структуры как векторы, двумерные - матрицы, трехмерные - кубы данных, массивы большей размерности - многомерные кубы данных.

Объявление многомерного массива в общем случае:

<тип>[, ... ,] <имя\_массива>;

Число запятых, увеличенное на единицу, и задает размерность массива. Следует отметить, что хотя явная инициализация с использованием многомерных константных массивов возможна, но применяется редко из-за громоздкости такой структуры. Проще инициализацию реализовать программно.

Примеры:  
Двумерный массив:

int[,] k = new int [2,3];

Обратите внимание, что пара квадратных скобок только одна. В нашем примере у массива 6 (=2\*3) элементов (k\*0,0+ - первый, k\*1,2+ - последний).

Аналогично мы можем задавать многомерные массивы. Пример трехмерного массива:

int[,,] k = new int [10,10,10];

Вариант инициализации многомерного массива:

int[,] k = {{2,-2},{3,-22},{0,4}};

Обратите внимение, что пара квадратных скобок только одна. В нашем примере у массива 6 (=2\*3) элементов (k\*0,0+ - первый, k\*1,2+ - последний).

Аналогично мы можем задавать многомерные массивы. Пример трехмерного массива:

int[,,] k = new int [10,10,10];

Вариант инициализации многомерного массива:

int[,] k = {{2,-2},{3,-22},{0,4}};

***Массивы массивов***

Еще одним видом массивов C# являются массивы массивов, называемые также изрезанными массивами (jagged arrays). Такой массив массивов можно рассматривать как одномерный массив, элементы которого являются массивами, элементы которых, в свою очередь, снова могут быть массивами, и так может продолжаться до некоторого уровня вложенности.

В каких ситуациях может возникать необходимость в таких структурах данных? 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. Пример:

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. Конечно, допустимо и такое объявление:

В этом случае элементы массива получат при инициализации нулевые значения. Реальную инициализацию нужно будет выполнять программным путем. Стоит заметить, что в конструкторе верхнего уровня константу 3 можно опустить и писать просто new int[ ][ ]. Вызов этого конструктора можно вообще опустить - он будет подразумеваться:

А вот конструкторы нижнего уровня необходимы. Еще одно важное замечание - динамические массивы возможны и здесь. В общем случае, границы на любом уровне могут быть выражениями, зависящими от переменных. Более того, допустимо, чтобы массивы на нижнем уровне были многомерными.

Еще несколько примеров:

//массив массивов

//объявление и инициализация

int[][] jagger = new int[3][]

{

new int[] {5,7,9,11},

new int[] {2,8},

new int[] {6,12,4}

};

int[][] jagger1 = new int[3][]

{

new int[4],

new int[2],

new int[3]

};

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. Конечно, допустимо и такое объявление:

int[][] jagger2 =

{

new int[4],

new int[2],

new int[3]

};

page3image3413675264

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. Конечно, допустимо и такое объявление:

//Объявляем 2-мерный ступенчатый массив

int[][] k = new int [2][];

//Объявляем 0-й элемент нашего ступенчатого массива

//Это опять массив и в нем 3 элемента

k[0]=new int[3];

//Объявляем 1-й элемент нашего ступенчатого массива

//Это опять массив и в нем 4 элемента

k[1]=new int[4];

k[1][3]=22; //записываем 22 в последний элемент массива ...

Обратите внимание, что у ступенчатых массивов задается несколько пар квадратных скобок (столько, сколько размерность у массива).

**Цикл foreach**

Новым видом цикла, который часто используется и достатточно удобен при работе с массивами, есть цикл foreach. Его синтаксис:

foreach(тип идентификатор in контейнер) оператор

Тело цикла выполняется для каждого элемента массива и заканчивается, когда полностью перебра- ны все элементы. Тип идентификатора должен быть согласован с типом элементов, хранящихся в массиве данных. Предполагается также, что элементы массива упорядочены. На каждом шаге цикла идентификатор, задающий текущий элемент массива, получает значение очередного элемента в соответствии с порядком, установленным на элементах массива. С этим текущим элементом и выполняется тело цикла - выполняется столько раз, сколько элементов находится в массиве.

Недостатком циклов foreach в языке C# является то, что цикл работает только на чтение, но не на запись элементов. Так что наполнять массив элементами приходится с помощью других операторов цикла.

Примеры использования цикла foreach:

class Program

{

static void Main(string[] args)

{

int[] array1 = {0, 2, 4, 6, 8, 10};

foreach (int n in array1)

{

System.Console.WriteLine(n.ToString()); }

string[] array2 = {"hello", "world"};

foreach (string s in array2)

{

System.Console.WriteLine(s);

}

}

}

В приведенном примере наш цикл перебирает все элементы массива array1. На это указывает строка

foreach(int n in array1)

которая интерпретируется так: для каждого целого числа из массива array1 делаем что-то там. Если бы элементами массива были бы не целые, а, скажем, вещественные числа, то запись выглядела бы так:

foreach(float n in array1)

Т.е. мы пишем именно тип элементов массива. Цикл foreach используется не только для массивов, но и для других объектов.

**Метод Format**

Каждый раз, когда выполнялся вывод результатов на консоль, неявно вызывался и метод Format. Общий синтаксис, специфицирующий формат, таков:

{N [,M [:<коды\_форматирования>]]}

Обязательный параметр N задает индекс объекта, заменяющего формат. Можно считать, что методу всегда передается массив объектов, даже если фактически передан один объект. Индексация объектов начинается с нуля, как это принято в массивах. Второй параметр M, если он задан, определяет минимальную ширину поля, которое отводится строке, вставляемой вместо формата. Третий необязательный параметр задает коды форматирования, указывающие, как следует форматировать объект. Например, код C (Currency) говорит о том, что параметр должен форматироваться как валюта с учетом национальных особенностей представления. Код P (Percent) задает форматирование в виде процентов с точностью до сотой доли.

Для вывода на консоль использовалась следующая конструкция:

Здесь мы используем внутри кавычек подстановочные знаки ,0-, ,1- и т. д. (нумерация в них идет с нуля). Переменные при этом выводятся в формате по умолчанию. Для вывода в определённом формате надо использовать подстановочные знаки с параметрами. Вот некоторые их них:

d - десятичный формат. Позволяет задать общее количество знаков (при необходимости число дополняется слева нулями).

f - формат с фиксированной точностью. Позволяет задать количество знаков после запятой.

x - шестнадцатеричный формат.

c - денежный формат (добавляет знак доллара и показывает два знака после запятой).

e - вывод числа в экспоненциальной форме.

Пример использования:

...  
 int x=23, y=-4;

...

Console.WriteLine("x={0}, y={1}", x, y); ...

...

int a=38;

//Выведется 0038

Console.WriteLine("a={0:d4}", a);

double pi=3.1415926;

//Выведется 3.14

Console.WriteLine("pi={0:f2}", pi);

int b=255;

//Выведется FF.

Console.WriteLine("b={0:X}", b);

int c=255;

//Выведется ff.

Console.WriteLine("c={0:x}", c);

double d=1003.214;

//Выведется $1, 003.14 в английской версии Windows и //1 003,14 р. в русской.

Console.WriteLine("d={0:c}", d);

double e=213.1;

//Выведется 2.131000e+002

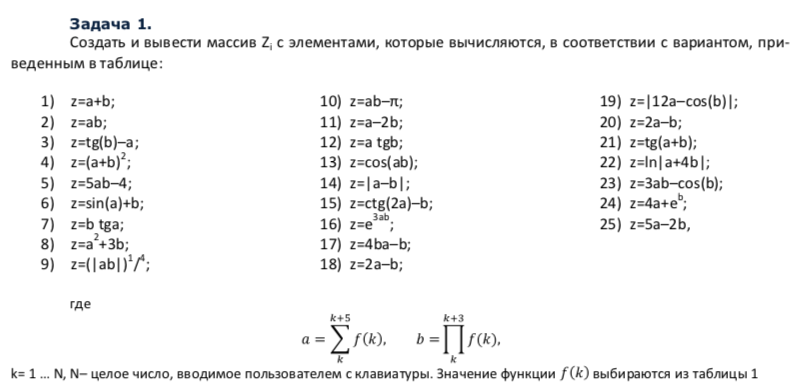
Console.WriteLine("e={0:e}", e);

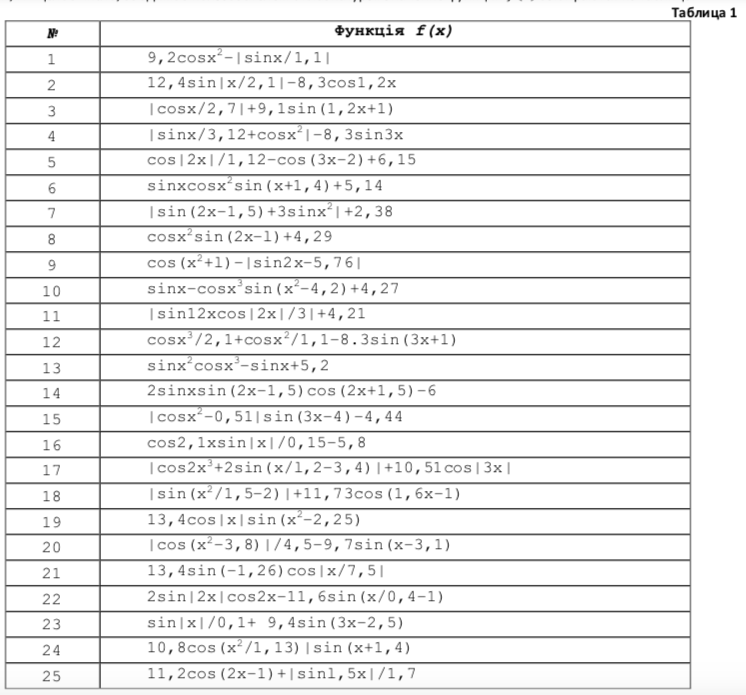
...

Параметры подстановочных знаков можно использовать как строчные, таки и прописные - это все равно. Исключение - вывод числа в шестнадцатеричном виде (при использовании h цифры a, ..., f будут строчными, при использовании H - прописными).

**ЗАДАНИЕ**

**Варианты индивидуальных заданий.**

******

******

В соответствии с вариантом, выполнить с полученным массивом следующее задание:

**Задание 1**

1. Создать и вывести на печать новый массив, который состоит с положительных элементов массива Z.
2. Определить сумму 3 и 6 положительных элементов.
3. Второй отрицательный элемент заменить минимальным.
4. Определить минимальный элемент среди положительных элементов.
5. Ненулевые элементы массива перенести в другой массив, расположив их в обратном порядке.
6. Вычислить сумму первых четырех отрицательных элементов.
7. Вывести номер предпоследнего положительного элемента.
8. Создать массив из отрицательных элементов массива.
9. Найти произведение второго и четвертого элементов больших 3.
10. Максимальный элемент поменять местами с минимальным элементом.
11. Последний отрицательный элемент поменять местами с максимальным элементом.
12. Вычислить произведение второго отрицательного и пятого элементов.
13. Сформировать другой массив из элементов массива, которые больше 1.

**Задание 2**

1. Вывести номера двух наибольших элементов массива. Вычислить их сумму.
2. Определить есть ли среди отрицательных элементов элементы с максимальными значениями.
3. Максимальный элемент поменять с 4 .
4. Третий положительный элемент заменить максимальным.
5. Определить номер пятого отрицательного элемента.
6. Вычислить произведение первых трех положительных элементов и определить их номера.
7. Вычислить сумму второго положительного и третьего элемента.
8. Создать новый массив из элементов меньших 3.
9. Создать массив, значения которого находятся между значениями 3 элемента и максимального значения.
10. Найти произведение номеров наименьших элементов среди положительных.
11. Определить сумму номеров 2 и 3 отрицательных элементов.
12. Определить номера 2 и 4 положительных элементов.