# Министерство образования Республики Беларусь

УПРАВЛЕНИЕ ОБРАЗОВАНИЯ МОГИЛЕВСКОГО ОБЛАСТНОГО ИСПОЛНИТЕЛЬНОГО КОМИТЕТА

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

«Могилевский государственный политехнический колледж»

# Домашняя контрольная работа №1

По дисциплине: «**Конструирование программ и языки программирования**»

Группа ПО-455

## Выполнил К.В.Инкин

Шифр 19

# 2021

**23 Перечислите приоритет операций**

В выражении с несколькими операторами операторы с более высоким приоритетом оцениваются до операторов с более низким приоритетом. В следующем примере умножение выполняется сначала, так как оно имеет более высокий приоритет, чем сложение:

var a = 2 + 2 \* 2;

Console.WriteLine(a); // output: 6

Чтобы изменить порядок вычисления, накладываемый приоритетом операторов используются скобки:

var a = (2 + 2) \* 2;

Console.WriteLine(a); // output: 8

В следующей таблице перечислены операторы C# в порядке убывания приоритета. Операторы в каждой строке имеют одинаковый приоритет. Приоритет операторов указан в таблице 1.

Таблица 1 – приоритет операторов.

| Операторы | Категория или имя |
| --- | --- |
| [x.y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#member-access-expression-), [f(x)](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#invocation-expression-), [a[i]](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#indexer-operator-), [x?.y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#null-conditional-operators--and-), [x?[y]](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#null-conditional-operators--and-), [x++](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#increment-operator-), [x--](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#decrement-operator---), [x!](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/null-forgiving), [new](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/new-operator), [typeof](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/type-testing-and-cast#typeof-operator), [checked](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/checked), [unchecked](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/keywords/unchecked), [default](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/default), [nameof](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/nameof), [delegate](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/delegate-operator), [sizeof](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/sizeof), [stackalloc](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/stackalloc), [x->y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/pointer-related-operators#pointer-member-access-operator--) | Первичный |
| [+x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#unary-plus-and-minus-operators), [-x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#unary-plus-and-minus-operators), [!x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#logical-negation-operator-), [~x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#bitwise-complement-operator-), [++x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#increment-operator-), [--x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#decrement-operator---), [^x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#index-from-end-operator-), [(T)x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/type-testing-and-cast#cast-expression), [await](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/await), [&x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/pointer-related-operators#address-of-operator-), [\*x](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/pointer-related-operators#pointer-indirection-operator-), [true и false](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/true-false-operators) | Унарный |
| [x..y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/member-access-operators#range-operator-) | Диапазон |
| [switch](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/switch-expression) | Выражение switch |
| [with](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/with-expression) | Выражение with |
| [x \* y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#multiplication-operator-), [x / y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#division-operator-), [x % y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#remainder-operator-) | Мультипликативный |
| [x + y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#addition-operator-), [x – y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#subtraction-operator--) | Аддитивный |
| [x << y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#left-shift-operator-), [x >> y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#right-shift-operator-) | Сдвиг |
| [x < y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/comparison-operators#less-than-operator-), [x > y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/comparison-operators#greater-than-operator-), [x <= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/comparison-operators#less-than-or-equal-operator-), [x >= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/comparison-operators#greater-than-or-equal-operator-), [is](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/type-testing-and-cast#is-operator), [as](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/type-testing-and-cast#as-operator) | Тестирование типов и относительный |
| [x == y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/equality-operators#equality-operator-), [x != y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/equality-operators#inequality-operator-) | Равенство |
| x & y | [Логическое И](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#logical-and-operator-) или [побитовое логическое И](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#logical-and-operator-) |
| x ^ y | [Логическое исключающее ИЛИ](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#logical-exclusive-or-operator-) или [побитовое логическое исключающее ИЛИ](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#logical-exclusive-or-operator-) |
| x | y | [Логическое ИЛИ](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#logical-or-operator-) или [побитовое логическое ИЛИ](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#logical-or-operator-) |
| [x && y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#conditional-logical-and-operator-) | Условное И |
| [x || y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#conditional-logical-or-operator-) | Условное ИЛИ |
| [x ?? y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/null-coalescing-operator) | Оператор объединения с NULL |
| [c ? t : f](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/conditional-operator) | Условный оператор |
| [x = y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/assignment-operator), [x += y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [x -= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [x \*= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [x /= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [x %= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/arithmetic-operators#compound-assignment), [x &= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [x |= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [x ^= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/boolean-logical-operators#compound-assignment), [x <<= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#compound-assignment), [x >>= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/bitwise-and-shift-operators#compound-assignment), [x ??= y](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/null-coalescing-operator), [=>](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/lambda-operator) | Назначение и объявление лямбда-выражений |

**57 Дайте определение понятию анонимного метода**

Анонимные методы используются для создания экземпляров делегатов.

Определение анонимных методов начинается с ключевого слова delegate, после которого идет в скобках список параметров и тело метода в фигурных скобках:

delegate(параметры)

{

// инструкции

}

Например:

class Program

{

delegate void MessageHandler(string message);

static void Main(string[] args)

{

MessageHandler handler = delegate(string mes)

{

Console.WriteLine(mes);

};

handler("hello world!");

Console.Read();

}

}

Анонимный метод не может существовать сам по себе, он используется для инициализации экземпляра делегата, как в данном случае переменная handler представляет анонимный метод. И через эту переменную делегата можно вызвать данный анонимный метод.

Другой пример анонимных методов - передача в качестве аргумента для параметра, который представляет делегат:

class Program

{

delegate void MessageHandler(string message);

static void Main(string[] args)

{

ShowMessage("hello!", delegate(string mes)

{

Console.WriteLine(mes);

});

Console.Read();

}

static void ShowMessage(string mes, MessageHandler handler)

{

handler(mes);

}

}

Если анонимный метод использует параметры, то они должны соответствовать параметрам делегата. Если для анонимного метода не требуется параметров, то скобки с параметрами опускаются. При этом даже если делегат принимает несколько параметров, то в анонимном методе можно вовсе опустить параметры:

class Program

{

delegate void MessageHandler(string message);

static void Main(string[] args)

{

MessageHandler handler = delegate

{

Console.WriteLine("анонимный метод");

};

handler("hello world!"); // анонимный метод

Console.Read();

}

}

То есть если анонимный метод содержит параметры, они обязательно должны соответствовать параметрам делегата. Либо анонимный метод вообще может не содержать никаких параметров, тогда он соответствует любому делегату, который имеет тот же тип возвращаемого значения.

При этом параметры анонимного метода не могут быть опущены, если один или несколько параметров определены с модификатором out.

Также, как и обычные методы, анонимные могут возвращать результат:

delegate int Operation(int x, int y);

static void Main(string[] args)

{

Operation operation = delegate (int x, int y)

{

return x + y;

};

int d = operation(4, 5);

Console.WriteLine(d);

Console.Read();

}

При этом анонимный метод имеет доступ ко всем переменным, определенным во внешнем коде:

delegate int Operation(int x, int y);

static void Main(string[] args)

{

int z = 8;

Operation operation = delegate (int x, int y)

{

return x + y + z;

};

int d = operation(4, 5);

Console.WriteLine(d);

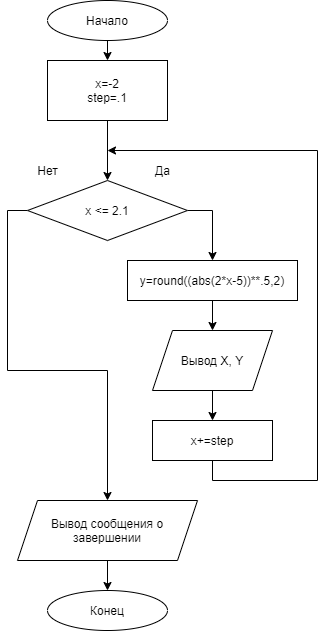
Console.Read();

}

В каких ситуациях используются анонимные методы? Когда нам надо определить однократное действие, которое не имеет много инструкций и нигде больше не используется. В частности, их можно использовать для обработки событий.

**73 Напишите программу, которая выводит на экран таблицу значений функции  в диапазоне от -2 до 2. Шаг изменения аргумента 0,1.**

**Блок-схема алгоритма решения задачи.**



**Текст программы.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp73

{

class Program

{

static void Main(string[] args)

{

double y;// значение функции

double x = -2;// начальное значение х

double step = .1;// шаг приращения

do

{

y = Math.Sqrt(Math.Abs(2\*x-5));// вычисляем

Console.WriteLine("x={0} y={1}", Math.Round(x,2), Math.Round(y,2));// выводим результат

x += step;// приращаем аргумент

}

while (x <= 2.1);// до конечного значения

Console.WriteLine("Работа завершена, нажмите ВВОД");

Console.Read();

}

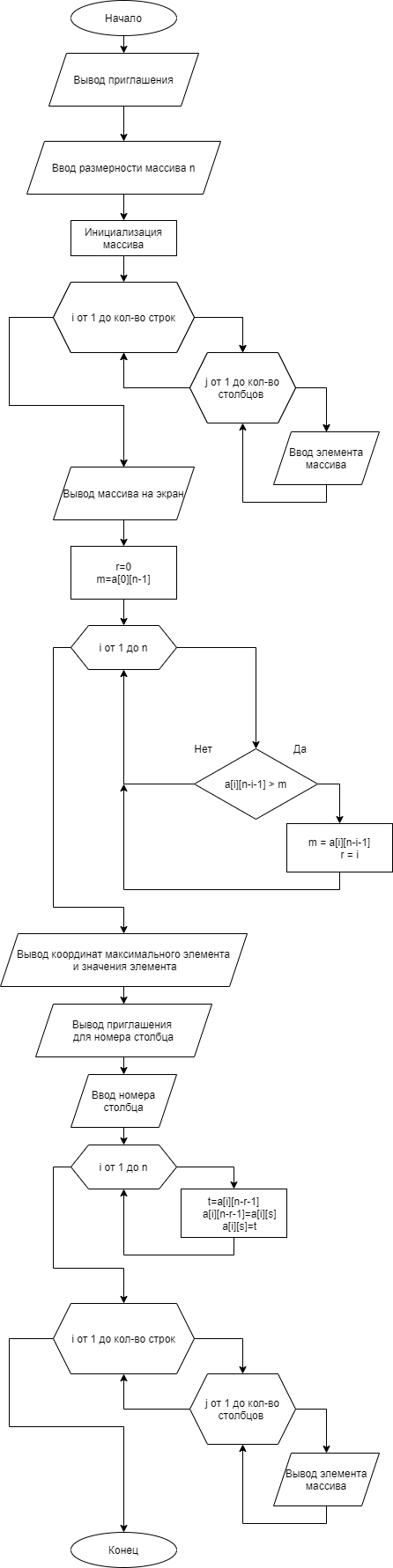
}

}

Результаты вычисленного значения функции и аргумент округлены до двух знаков после запятой для более эстетичного вывода.

**90 Решите задачу. Дан двумерный массив n х n. Заполните его по строкам с клавиатуры. Определите максимальный элемент в побочной диагонали и поменяйте местами столбец содержащего его с первым столбцом.**

**Блок-схема алгоритма решения задачи.**



**Текст программы.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp90

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Введите размер массива n:");

int n = Int32.Parse(Console.ReadLine());

int[,] a = new int[n, n];//объявление массива

Console.WriteLine("Размер масива: [{0}x{1}]", n, n);

for (int i = 0; i < n; i++)// считываем массив

{

for (int j = 0; j < n; j++)

{

Console.WriteLine("Заполните элемент массива A[{0},{1}]", i + 1, j + 1);

a[i, j] = Int32.Parse(Console.ReadLine());

}

}

// вывод массива

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

{// перебираем столбцы

for (int j = 0; j < n; j++)

{// перебираем элементы столбца

Console.Write(a[i, j]);

Console.Write(" ");

}

Console.WriteLine("");

}

// поиск столбца с максимальным элементом

int r = 0;

int m = a[0, n - 1];

for (int i = 0; i < n; i++) {//цикл по элементам

if (a[i, n - i - 1] > m) {// сравниваем элемент побочной диагонали с максимальным

m = a[i,n - i - 1];// обновляем максимальный

r = i;// запоминаем столбец

}

}

// выводим промежуточный результат

// и запросим номер столбца обмена

Console.WriteLine("Координаты максимального элемента i: {0} j:{1}", r + 1, n - r);

Console.WriteLine("Максимальный элемент: {0}", m);

Console.WriteLine("Номер столбца c которым поменять столбец {0}:", n - r);

int s = Int32.Parse(Console.ReadLine())-1; // считываем номер столбца

// совершаем сам обмен

int t = 0;// временная переменная обмена

for (int i = 0; i < n; i++) {

t = a[i, n - r - 1];

a[i, n - r - 1] = a[i, s];

a[i, s] = t;

}

// вывод результирующего массива

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

{// перебираем столбцы

for (int j = 0; j < n; j++)

{// перебираем элементы столбца

Console.Write(a[i, j]);

Console.Write(" ");

}

Console.WriteLine("");

}

Console.WriteLine("Нажмите ВВОД");

Console.ReadLine();

}

}

}

**Список использованных источников**

1. Албахари, Д. C# 5.0. Справочник. Полное описание языка / Д.Албахари. – М.: Вильямс, 2014.
2. Лабор, В.В. Си Шарп: Создание приложений для Windows / В.В.Лабор. - Мн.: Харвест, 2003.
3. Павловская, Т.А С#. Программирование на языке высокого уровня / Т.А.Павловская. - СПб: Питер, 2014.
4. Фролов, А.В. Визуальное проектирование приложений С# / А.В.Фролов. - М: КУДИЦ - ОБРАЗ, 2003.
5. Фленов, М. Библия С# / М.Фленов. - СПб.: Питер, 2011.