Министерство науки и высшего образования РФ
ФГАОУ ВПО
Национальный исследовательский технологический университет «МИСиС»

Институт Информационных технологий и компьютерных наук (ИТКН)

Кафедра Инфокоммуникационных технологий (ИКТ)

Отчет по лабораторной работе №5
по дисциплине «Программирование и Алгоритмизация»
на тему «Методы»

Выполнил:
студент группы БИВТ-24-5

Черных Богдан

Проверил:
Стучилин В. В.

Москва, 2024

Теоретическое введение. Метод – это последовательность инструкций (операторов) для решения какой-либо более или менее самостоятельной задачи (подзадачи), оформленная специальным образом. Первое знакомство со стандартными методами у вас уже состоялось при использовании метода `Main`, а также некоторых методов класса `Console` (`ReadLine()`, `WriteLine()` и др.), класса `Math` (`Abs()`, `Sin()` и др.) и др.

Сейчас мы рассмотрим методы более последовательно и подробно и научимся создавать их самостоятельно.

Фактически, в языке C# все алгоритмы обработки данных являются методами (например, все предыдущие программы создавались в рамках метода `Main`).

Методы в некотором смысле можно рассматривать в качестве аналогов подпрограмм (процедур и функций) в языках процедурного программирования, однако там данные и подпрограммы их обработки формально не связаны. В ООП же методы непосредственно связаны с объектом и определяют действия, которые можно выполнять над объектом или которые сам объект может выполнять.

Использование методов:

- улучшает структуру программы и облегчает ее восприятие;
- облегчает отладку, так как каждый метод может отлаживаться отдельно;
- при решении некоторой подзадачи несколько раз в рамках одной программы позволяет оформлять алгоритм один раз, а обращаться к нему многократно;
- уменьшает объем программы, если один и тот же метод выполняется несколько раз. Время выполнения при этом практически не изменяется.

Метод обычно реализует алгоритм в достаточно общем виде с использованием *формальных параметров* – абстрактных обозначений величин, участвующих в описании алгоритма. Каждый метод имеет свое имя, по которому осуществляется обращение к нему. При вызове метода формальные параметры заменяются на *фактические* (конкретизирующие задачу), с которыми метод и выполняется.

Формальным параметром может быть: переменная, массив, а также другой метод (типа делегат). Перед именем формального параметра указывается его тип (`int`, `double`, `string` и т. д.). Если формальным параметром является массив, то за типом массива указывается квадратные скобки (для одномерного массива), либо в квадратных скобках ставится , (запятая) (для двухмерного массива). Формальные параметры показывают, какие операции и над какими

данными необходимо выполнять для реализации алгоритма. Фактическим параметром может быть литерал, переменная, выражение, массив, метод. Фактические параметры содержат данные, над которыми выполняются операции. Фактические параметры должны соответствовать формальным по количеству, типу и порядку следования.

Формальные параметры бывают входными и выходными. Через входные параметры передаются данные для вычисления алгоритма, выходные параметры содержат результат выполнения алгоритма (метода). Ключевое слово `out` или `ref` перед выходным параметром означает передачу параметра по ссылке, т.е. при обращении к методу на место выходного параметра передается адрес аргумента, фигурирующего в обращении к методу.

Переменные, не являющиеся ни входными, ни выходными аргументами, являются внутренними переменными метода и в списке параметров не указываются.

Алгоритм выполняется лишь при вызове метода.

З а м е ч а н и е . Взаимное расположение методов в пределах одного класса не имеет значения. Управление всегда вначале передается методу `Main`.

Цель: освоение навыков разработки и использования методов в языке программирования C#. В процессе выполнения работы студенты научатся создавать собственные методы, структурировать программы с их помощью, а также применять делегаты для передачи функций в методы. Особое внимание уделяется использованию методов для организации кода, улучшения его читаемости и упрощения отладки. Работа включает разработку методов для обработки данных, выполнения арифметических расчетов и работы с массивами и матрицами, а также знакомит с основными принципами объектно-ориентированного программирования (ООП) и применения делегатов для выполнения задач в общем виде.

Код заданий:

```
//svg does precious
using System;                            // Аналог <iostream> для работы с консолью
и основными функциями
using System.Collections.Generic;        // Аналог <vector>, <list>, <map>, <set>,
<unordered_map>, <unordered_set>, <stack>, <queue>
using System.Text;                       // Аналог <string>, <cstring> (для работы
со строками и StringBuilder)
using System.Linq;                       // Аналог <algorithm> (для работы с LINQ,
сортировок, поиска и т.д.)
using System.IO;                         // Аналог <cstdio>, <fstream> (работа с
файлами)
using System.Globalization;              // Аналог <iomanip> (для форматирования)
using System.Collections;                // Работа с различными коллекциями
(например, ArrayList)
using System.Threading;                  // Потоки и многопоточность
using System.Runtime.Serialization;      // Аналог <stdexcept> (работа с
исключениями)
```

```
using System.Reflection;                // Аналог <typeinfo> (информация о типах,
рефлексия)
using System.Diagnostics;               // Аналог <utility>, <std::pair>
(вспомогательные функции и классы)
using System.ComponentModel;            // Дополнительные утилиты и атрибуты
using System.Numerics;                  // Работа с большими числами и
математическими операциями
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
// Для работы с потоками данных:
using System.Threading.Tasks;           // Асинхронные задачи

// Для работы с датами и временем:
using System.Timers;                    // Для работы с таймерами и временем
using System.Collections.Generic;
using System.Text;
using System.Linq;
using System.IO; //important
using System.Globalization;
using System.Collections;
using System.Threading;
using System.Runtime.Serialization;
using System.Reflection;
using System.Diagnostics;
using System.ComponentModel;
using System.Numerics;
using System.Globalization;
using System.Diagnostics;
using System.Net;
using System.Numerics;
using System.Threading.Tasks;
using System.Security.Cryptography;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
using System.Xml.Linq;
using System.Runtime.InteropServices;
using System.Security;
using System.Web;
using System.Media;
using System.Drawing;
using System.Configuration;
using System.Timers;
using System.Runtime.Remoting;
using System.Runtime.CompilerServices;
using System.Runtime.Versioning;
using System.CodeDom;
using System.CodeDom.Compiler;
using System.Collections.Concurrent;
using System.Runtime;
using System.Windows;
using System.Windows.Input;
using System.Security.Principal;
using System.Security.Permissions;
using System.Resources;
using C = System.Console; //console
using dl = System.Decimal;//decimal
using str = System.String;//string
using l = System.Int64;   //long
using u = System.UInt64;  //Ulong
```

```
using db = System.Double; //Double

/*
Izzspot - 19 years
Boogie B - 20 years
SJ - 21 years
Bandokay - life sentence

Youngest in the charge
OFB, we don't window shop
Bro caught him an opp and tried turn him off (Bow, bow)
In this X3, man's swervin' off (Skrr, skrr)
Free Boogie Bando, he got birded off (Free my bro, free my bro)
Whenever we get a burner loss
We just cop a next one and go burst it off (Ay)
Lil bro's tellin' me he got his earnings wrong
We just took him OT, now his trapline's gone (Ring, ring)
Hashtag
Bro backed this ting and just started squeezin' (Clarted)
When it broad day, it was freezin'
Hashtag fuckery, hashtag screamin'
One on the hand ting woi, left man leanin', leanin' (Fucker)
Show us cause it's good to feel it
Shortie's cooze and she must be dreamin'
*/
/* ¡Adelante Barcelona, adelante Cataluña! Visca el Barça! Visca Catalunya!
   ¡Al diablo con todos los demás, porque lo más importante en la vida es el
fútbol!
*/
//--------------------------------------------------------------------------------
-----
//--------------------------------------------------------------------------------
-----
//--------------------------------------------------------------------------------
-----



class Program
{//Лабораторная работа №5
 // 1 Уровень 1 Задача, 1 Уровень 2 Задача, 1 Уровень 3 Задача,
 // 2 Уровень 25 Задача, 2 Уровень 26 Задача, 2 Уровень 27 Задача,
 // 3 Уровень 1 Задача, 3 Уровень 2 Задача, 3 Уровень 3 Задача
    static void Main()
    {
        Console.WriteLine("1 УРОВЕНЬ 1 ЗАДАЧА");
        Task11.Run();

        Console.WriteLine("\n1 УРОВЕНЬ 2 ЗАДАЧА");
        Task12.Run();

        Console.WriteLine("\n1 УРОВЕНЬ 3 ЗАДАЧА");
        Task13.Run();

        Console.WriteLine("\n2 УРОВЕНЬ 25 ЗАДАЧА");
        Task225.Run();

        Console.WriteLine("\n2 УРОВЕНЬ 26 ЗАДАЧА");
        Task226.Run();

        Console.WriteLine("\n2 УРОВЕНЬ 27 ЗАДАЧА");
        Task227.Run();
```

```
        Console.WriteLine("\n3 УРОВЕНЬ 1 ЗАДАЧА");
        Task31.Run();

        Console.WriteLine("\n3 УРОВЕНЬ 2 ЗАДАЧА");
        Task32.Run();

        Console.WriteLine("\n3 УРОВЕНЬ 3 ЗАДАЧА");
        Task33.Run();
    }
}

class Task11
{
    public static void Run()
    {
        Console.WriteLine("Количество способов отбора команды:");
        Console.WriteLine("Из 8 кандидатов: " + CalculateCombinations(8, 5));
        Console.WriteLine("Из 10 кандидатов: " + CalculateCombinations(10, 5));
        Console.WriteLine("Из 11 кандидатов: " + CalculateCombinations(11, 5));
    }

    private static int CalculateCombinations(int n, int k)
    {
        return Factorial(n) / (Factorial(k) * Factorial(n - k));
    }

    private static int Factorial(int x)
    {
        int result = 1;
        for (int i = 2; i <= x; i++)
        {
            result *= i;
        }
        return result;
    }
}

class Task12
{
    public static void Run()
    {
        //Console.WriteLine("Введите стороны первого треугольника (a, b, c):");
        double a1 = 3; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона a1 " + a1);
        double b1 = 4; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона b1 " + b1);
        double c1 = 5; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона c1 " + c1);

        //Console.WriteLine("Введите стороны второго треугольника (a, b, c):");
        double a2 = 4; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона a2 " + a2);
        double b2 = 5; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона b2 " + b2);
        double c2 = 6; //Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Сторона c2 " + c2);

        string result = CompareTrianglesByArea(a1, b1, c1, a2, b2, c2);
        Console.WriteLine(result);
    }
```

```
    private static string CompareTrianglesByArea(double a1, double b1, double c1,
double a2, double b2, double c2)
    {
        double area1 = CalculateTriangleArea(a1, b1, c1);
        double area2 = CalculateTriangleArea(a2, b2, c2);

        if (area1 > area2)
            return "Первый треугольник имеет большую площадь.";
        else if (area2 > area1)
            return "Второй треугольник имеет большую площадь.";
        else
            return "Треугольники имеют равные площади.";
    }

    private static double CalculateTriangleArea(double a, double b, double c)
    {
        double p = (a + b + c) / 2;
        return Math.Sqrt(p * (p - a) * (p - b) * (p - c));
    }
}

class Task13
{
    public static void Run()
    {
        Console.WriteLine("Пройденное расстояние через 1 час:");
        Console.WriteLine("Первый велосипедист: " + CalculateDistanceAfterTime(10,
1, 1));
        Console.WriteLine("Второй велосипедист: " + CalculateDistanceAfterTime(9,
1.6, 1));

        Console.WriteLine("\nПройденное расстояние через 4 часа:");
        Console.WriteLine("Первый велосипедист: " + CalculateDistanceAfterTime(10,
1, 4));
        Console.WriteLine("Второй велосипедист: " + CalculateDistanceAfterTime(9,
1.6, 4));

        double catchUpTime = CalculateCatchUpTime(10, 1, 9, 1.6);
        Console.WriteLine("\nВремя, когда второй догонит первого: " + catchUpTime
+ " часов");
    }

    private static double CalculateDistanceAfterTime(double initialVelocity,
double acceleration, double time)
    {
        return initialVelocity * time + 0.5 * acceleration * time * time;
    }

    private static double CalculateCatchUpTime(double v1, double a1, double v2,
double a2)
    {
        return (v2 - v1) / (a1 - a2);
    }
}


class Task225
{
    public static void Run()
    {
        int[,] matrix1 = { { -3, 5, -1 }, { -2, -4, 6 }, { 3, -2, -5 } };
        int[,] matrix2 = { { 2, -1, -3 }, { -4, -1, -7 }, { 0, 4, -8 } };
```

```
        Console.WriteLine("Исходная матрица 1:");
        PrintMatrix(matrix1);

        Console.WriteLine("Исходная матрица 2:");
        PrintMatrix(matrix2);

        int maxNegativesRow1 = FindRowWithMostNegatives(matrix1);
        int maxNegativesRow2 = FindRowWithMostNegatives(matrix2);

        Console.WriteLine("Строка с максимальным количеством отрицательных
элементов в первой матрице: " + maxNegativesRow1);
        Console.WriteLine("Строка с максимальным количеством отрицательных
элементов во второй матрице: " + maxNegativesRow2);
    }

    private static int FindRowWithMostNegatives(int[,] matrix)
    {
        int maxNegativesCount = -1;
        int rowWithMaxNegatives = -1;

        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            int negativesCount = CountNegativesInRow(matrix, i);
            if (negativesCount > maxNegativesCount)
            {
                maxNegativesCount = negativesCount;
                rowWithMaxNegatives = i;
            }
        }

        return rowWithMaxNegatives;
    }

    private static int CountNegativesInRow(int[,] matrix, int row)
    {
        int count = 0;
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            if (matrix[row, j] < 0)
                count++;
        }
        return count;
    }
    private static void PrintMatrix(int[,] matrix)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write(matrix[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}

class Task226
{
    public static void Run()
    {
        int[,] matrix1 = { { -3, 5, -1 }, { -2, -4, 6 }, { -3, -2, -5 } };
```

```
        int[,] matrix2 = { { 2, -1, -3 }, { -4, -1, -7 }, { 0, 4, -8 } };
        Console.WriteLine("Исходная матрица 1:");
        PrintMatrix(matrix1);

        Console.WriteLine("Исходная матрица 2:");
        PrintMatrix(matrix2);

        int row1 = FindRowWithMostNegatives(matrix1);
        int row2 = FindRowWithMostNegatives(matrix2);

        if (row1 != -1 && row2 != -1)
        {
            SwapRows(matrix1, matrix2, row1, row2);

            Console.WriteLine("Матрица 1 после обмена строк:");
            PrintMatrix(matrix1);

            Console.WriteLine("Матрица 2 после обмена строк:");
            PrintMatrix(matrix2);
        }
        else
        {
            Console.WriteLine("Обмен невозможен: строки с максимальным количеством
отрицательных элементов не найдены.");
        }
    }

    private static int FindRowWithMostNegatives(int[,] matrix)
    {
        int maxNegativesCount = -1;
        int rowWithMaxNegatives = -1;

        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            int negativesCount = CountNegativesInRow(matrix, i);
            if (negativesCount > maxNegativesCount)
            {
                maxNegativesCount = negativesCount;
                rowWithMaxNegatives = i;
            }
        }

        return rowWithMaxNegatives;
    }

    private static int CountNegativesInRow(int[,] matrix, int row)
    {
        int count = 0;
        for (int j = 0; j < matrix.GetLength(1); j++)
        {
            if (matrix[row, j] < 0)
                count++;
        }
        return count;
    }

    private static void SwapRows(int[,] matrix1, int[,] matrix2, int row1, int
row2)
    {
        for (int j = 0; j < matrix1.GetLength(1); j++)
        {
            int temp = matrix1[row1, j];
```

```
            matrix1[row1, j] = matrix2[row2, j];
            matrix2[row2, j] = temp;
        }
    }

    private static void PrintMatrix(int[,] matrix)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write(matrix[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}


class Task227
{
    public static void Run()
    {
        int[,] matrix1 = { { 2, -1, 4 }, { 3, -4, -7 }, { 5, -2, 8 } };
        int[,] matrix2 = { { -6, 3, -2 }, { 8, 1, -5 }, { 7, -3, 2 } };

        Console.WriteLine("Исходная матрица 1:");
        PrintMatrix(matrix1);

        Console.WriteLine("Исходная матрица 2:");
        PrintMatrix(matrix2);

        ProcessMatrix(matrix1);
        ProcessMatrix(matrix2);

        Console.WriteLine("Матрица 1 после обработки:");
        PrintMatrix(matrix1);

        Console.WriteLine("Матрица 2 после обработки:");
        PrintMatrix(matrix2);
    }

    private static void ProcessMatrix(int[,] matrix)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            int maxIndex = FindMaxInRow(matrix, i);
            if (i % 2 == 0)
            {
                matrix[i, maxIndex] = 0; // Для четных строк заменяем на 0
            }
            else
            {
                matrix[i, maxIndex] *= (maxIndex + 1); // Для нечетных строк
умножаем на номер столбца (maxIndex + 1)
            }
        }
    }

    private static int FindMaxInRow(int[,] matrix, int row)
    {
        int maxIndex = 0;
```

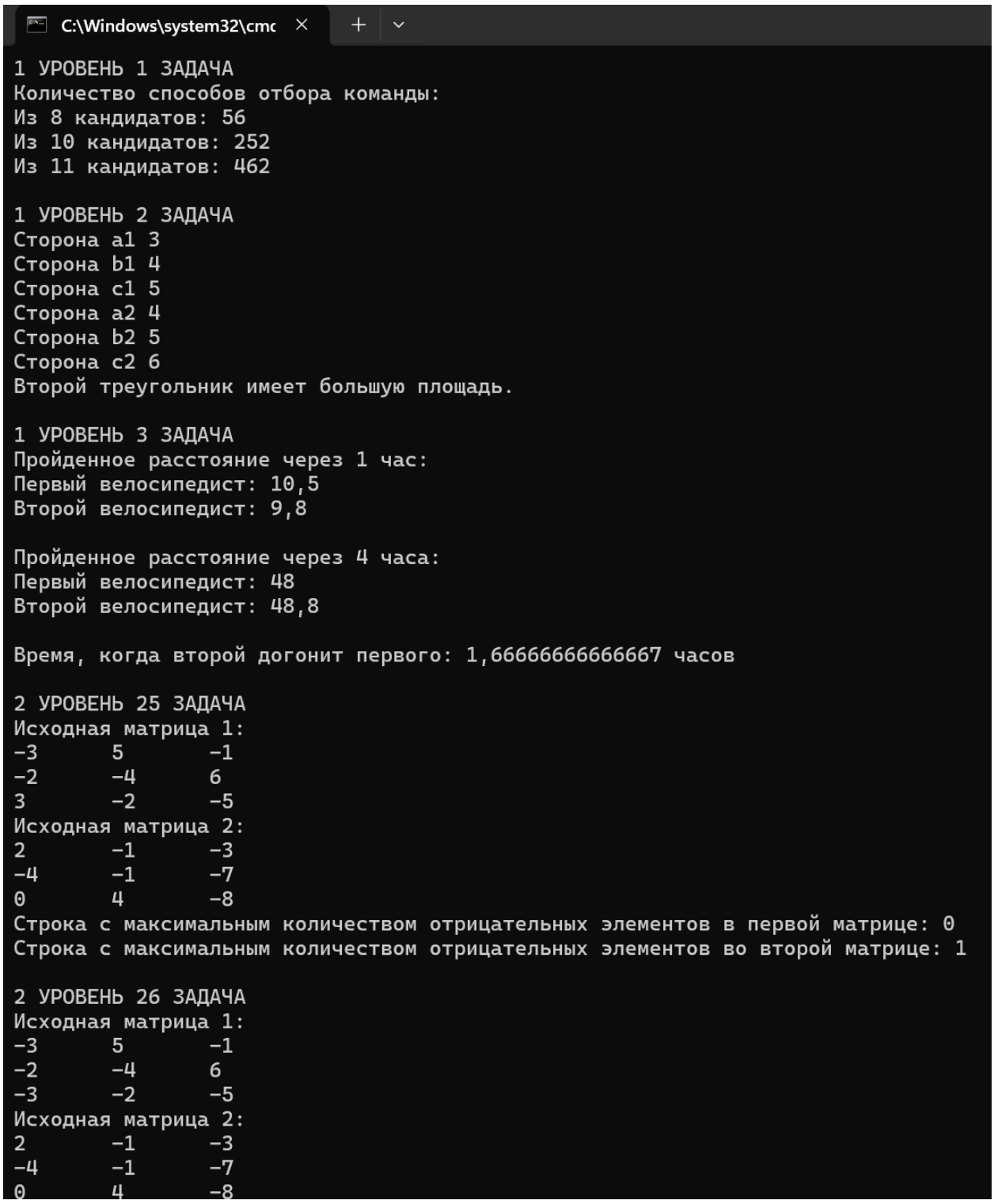
```
        for (int j = 1; j < matrix.GetLength(1); j++)
        {
            if (matrix[row, j] > matrix[row, maxIndex])
            {
                maxIndex = j;
            }
        }
        return maxIndex;
    }

    private static void PrintMatrix(int[,] matrix)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write(matrix[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}

class Task31
{
    public delegate double SumTerm(double x, int i);

    public static void Run()
    {
        double a1 = 0, b1 = 1, h1 = 0.1;
        double a2 = Math.PI / 5, b2 = Math.PI, h2 = Math.PI / 25;

        CalculateAndPrintSum(a1, b1, h1, SumTerm1);
        CalculateAndPrintSum(a2, b2, h2, SumTerm2);
    }

    private static void CalculateAndPrintSum(double a, double b, double h, SumTerm
termFunc)
    {
        for (double x = a; x <= b; x += h)
        {
            double sum = 0;
            int i = 1;
            double term;
            do
            {
                term = termFunc(x, i);
                sum += term;
                i++;
            } while (Math.Abs(term) > 1e-6);  // условие выхода по точности
            Console.WriteLine($"Сумма при x = {x:F2}: {sum:F4}");
        }
    }

    private static double SumTerm1(double x, int i)
    {
        return Math.Cos(i * x) / Factorial(i);
    }

    private static double SumTerm2(double x, int i)
    {
        return Math.Pow(-1, i) * Math.Cos(i * x) / (i * i);
```

```
    }

    private static double Factorial(int n)
    {
        double result = 1;
        for (int i = 2; i <= n; i++)
        {
            result *= i;
        }
        return result;
    }
}


class Task32
{
    public delegate void SortDelegate(int[] row);

    public static void Run()
    {
        int[,] matrix = {
            { 3, 5, 1, 8 },
            { 7, 2, 4, 6 },
            { 9, 3, 5, 1 },
            { 4, 8, 6, 2 }
        };

        Console.WriteLine("Исходная матрица:");
        PrintMatrix(matrix);

        ProcessMatrix(matrix, SortAscending, SortDescending);

        Console.WriteLine("Матрица после обработки:");
        PrintMatrix(matrix);
    }

    private static void ProcessMatrix(int[,] matrix, SortDelegate sortEven,
SortDelegate sortOdd)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            int[] row = GetRow(matrix, i);
            if (i % 2 == 0)
                sortEven(row);
            else
                sortOdd(row);
            SetRow(matrix, i, row);
        }
    }

    private static int[] GetRow(int[,] matrix, int rowIndex)
    {
        int[] row = new int[matrix.GetLength(1)];
        for (int j = 0; j < row.Length; j++)
        {
            row[j] = matrix[rowIndex, j];
        }
        return row;
    }

    private static void SetRow(int[,] matrix, int rowIndex, int[] row)
    {
```

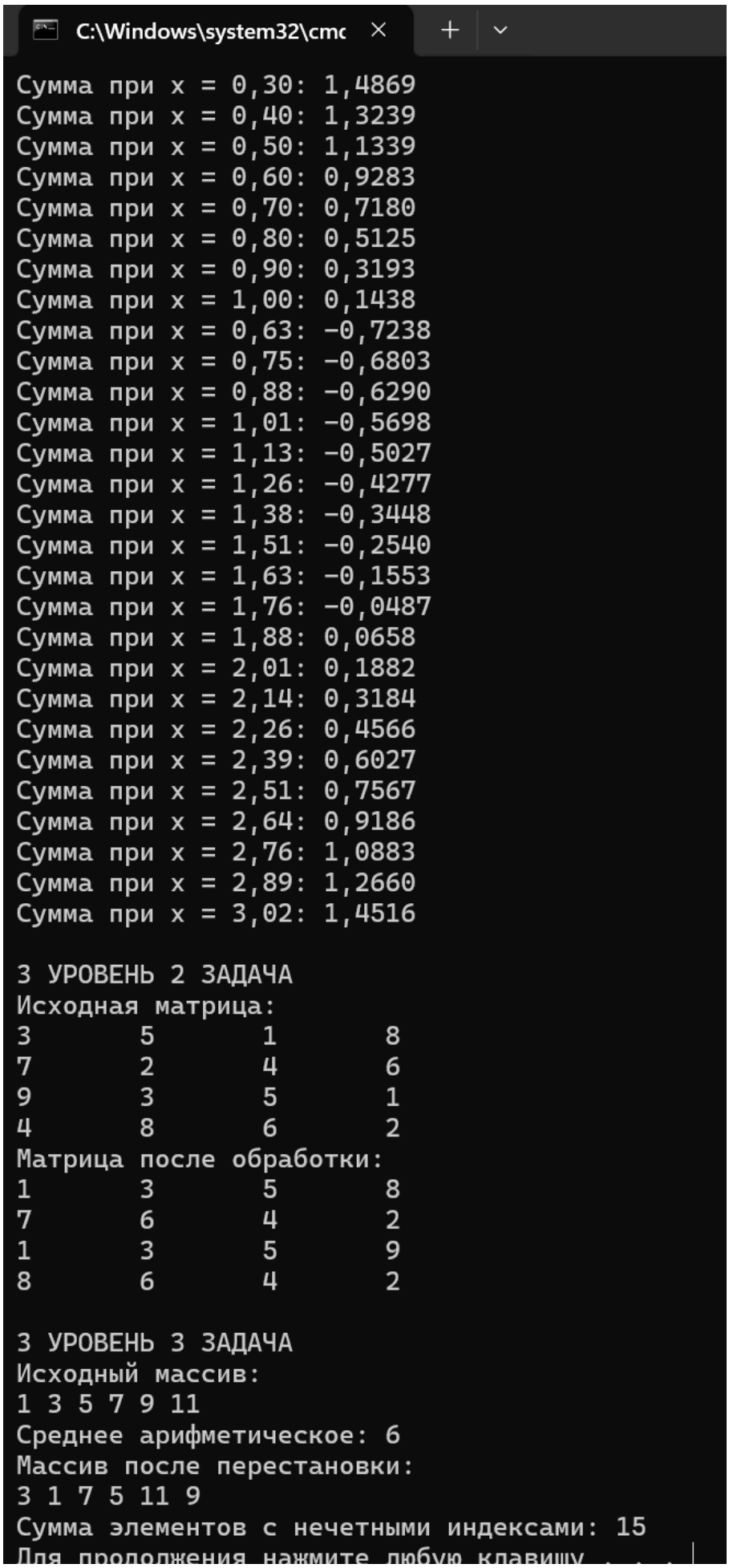
```
        for (int j = 0; j < row.Length; j++)
        {
            matrix[rowIndex, j] = row[j];
        }
    }

    private static void SortAscending(int[] row)
    {
        Array.Sort(row);
    }

    private static void SortDescending(int[] row)
    {
        Array.Sort(row);
        Array.Reverse(row);
    }

    private static void PrintMatrix(int[,] matrix)
    {
        for (int i = 0; i < matrix.GetLength(0); i++)
        {
            for (int j = 0; j < matrix.GetLength(1); j++)
            {
                Console.Write(matrix[i, j] + "\t");
            }
            Console.WriteLine();
        }
    }
}

class Task33
{
    public delegate void SwapDelegate(int[] array);

    public static void Run()
    {
        int[] array = { 1, 3, 5, 7, 9, 11 };
        Console.WriteLine("Исходный массив:");
        PrintArray(array);

        double average = CalculateAverage(array);
        Console.WriteLine("Среднее арифметическое: " + average);

        SwapDelegate swapMethod;
        if (array[0] > average)
        {
            swapMethod = SwapStartingFromFirst;
        }
        else
        {
            swapMethod = SwapStartingFromLast;
        }
        swapMethod(array);

        Console.WriteLine("Массив после перестановки:");
        PrintArray(array);

        int sum = CalculateSumOfOddIndexedElements(array);
        Console.WriteLine("Сумма элементов с нечетными индексами: " + sum);
    }

    private static double CalculateAverage(int[] array)
```

```
    {
        double sum = 0;
        foreach (int item in array)
        {
            sum += item;
        }
        return sum / array.Length;
    }

    private static void SwapStartingFromFirst(int[] array)
    {
        for (int i = 0; i < array.Length - 1; i += 2)
        {
            int temp = array[i];
            array[i] = array[i + 1];
            array[i + 1] = temp;
        }
    }

    private static void SwapStartingFromLast(int[] array)
    {
        for (int i = array.Length - 1; i > 0; i -= 2)
        {
            int temp = array[i];
            array[i] = array[i - 1];
            array[i - 1] = temp;
        }
    }

    private static int CalculateSumOfOddIndexedElements(int[] array)
    {
        int sum = 0;
        for (int i = 1; i < array.Length; i += 2)
        {
            sum += array[i];
        }
        return sum;
    }

    private static void PrintArray(int[] array)
    {
        foreach (int item in array)
        {
            Console.Write(item + " ");
        }
        Console.WriteLine();
    }
}
```

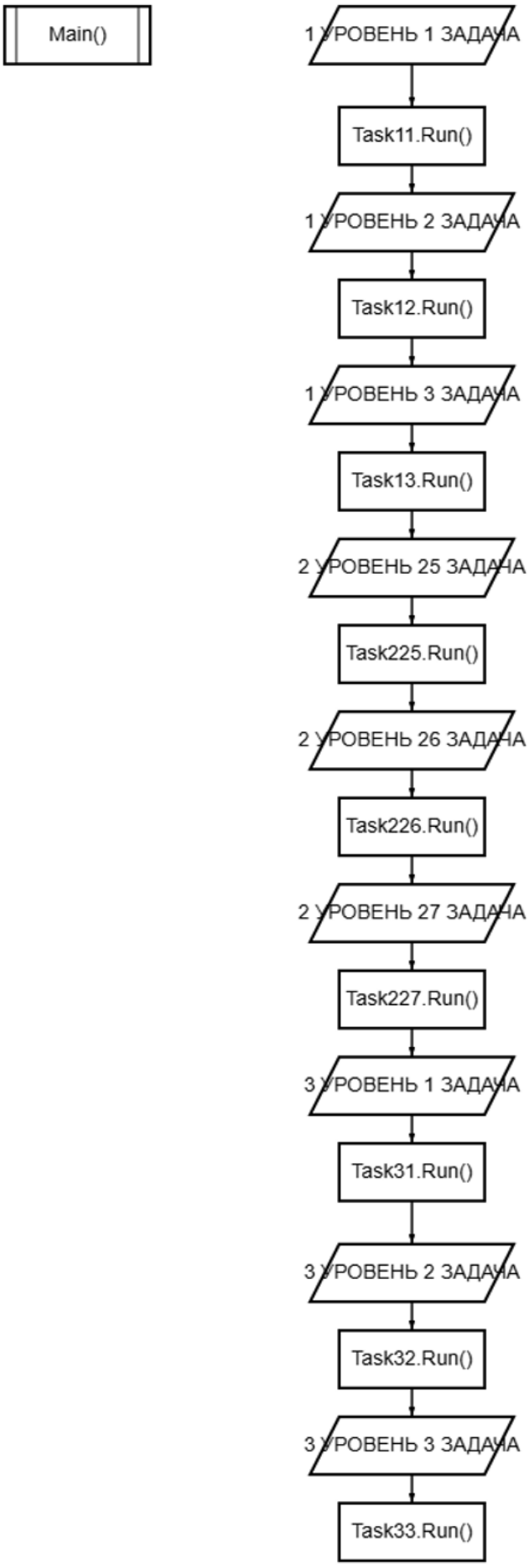
Выполнение программы:


C:\Windows\system32\cmc
1 УРОВЕНЬ 1 ЗАДАЧА
Количество способов отбора команды:
Из 8 кандидатов: 56
Из 10 кандидатов: 252
Из 11 кандидатов: 462
1 УРОВЕНЬ 2 ЗАДАЧА
Сторона a1 3
Сторона b1 4
Сторона c1 5
Сторона a2 4
Сторона b2 5
Сторона c2 6
Второй треугольник имеет большую площадь.
1 УРОВЕНЬ 3 ЗАДАЧА
Пройденное расстояние через 1 час:
Первый велосипедист: 10,5
Второй велосипедист: 9,8
Пройденное расстояние через 4 часа:
Первый велосипедист: 48
Второй велосипедист: 48,8
Время, когда второй догонит первого: 1,66666666666667 часов
2 УРОВЕНЬ 25 ЗАДАЧА
Исходная матрица 1:
-3 5 -1
-2 -4 6
3 -2 -5
Исходная матрица 2:
2 -1 -3
-4 -1 -7
0 4 -8
Строка с максимальным количеством отрицательных элементов в первой матрице: 0
Строка с максимальным количеством отрицательных элементов во второй матрице: 1
2 УРОВЕНЬ 26 ЗАДАЧА
Исходная матрица 1:
-3 5 -1
-2 -4 6
-3 -2 -5
Исходная матрица 2:
2 -1 -3
-4 -1 -7
0 4 -8

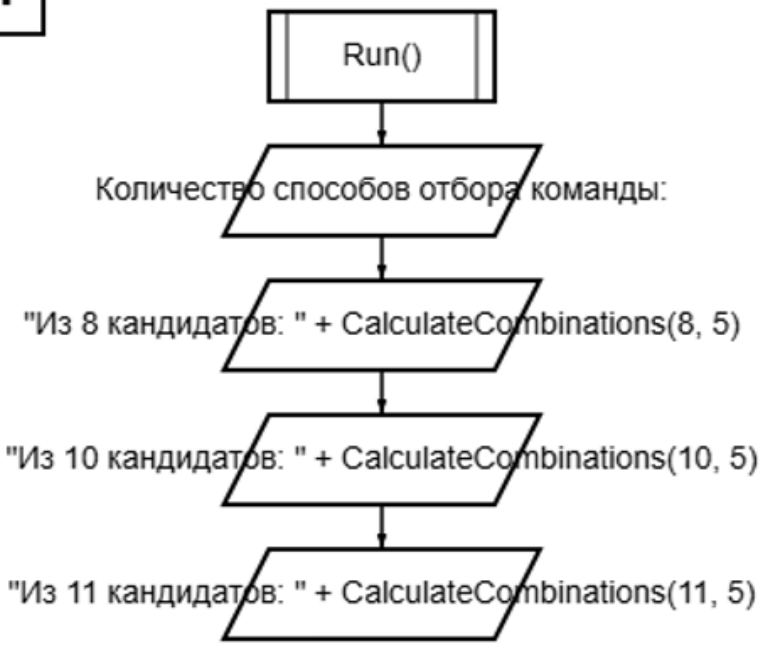
```
Исходная матрица 2:
2       -1      -3
-4      -1      -7
0       4       -8
Матрица 1 после обмена строк:
-3      5       -1
-2      -4      6
-4      -1      -7
Матрица 2 после обмена строк:
2       -1      -3
-3      -2      -5
0       4       -8

2 УРОВЕНЬ 27 ЗАДАЧА
Исходная матрица 1:
2       -1      4
3       -4      -7
5       -2      8
Исходная матрица 2:
-6      3       -2
8       1       -5
7       -3      2
Матрица 1 после обработки:
2       -1      0
3       -4      -7
5       -2      0
Матрица 2 после обработки:
-6      0       -2
8       1       -5
0       -3      2

3 УРОВЕНЬ 1 ЗАДАЧА
Сумма при x = 0,00: 1,7183
Сумма при x = 0,10: 1,6913
Сумма при x = 0,20: 1,6122
Сумма при x = 0,30: 1,4869
Сумма при x = 0,40: 1,3239
Сумма при x = 0,50: 1,1339
Сумма при x = 0,60: 0,9283
Сумма при x = 0,70: 0,7180
Сумма при x = 0,80: 0,5125
Сумма при x = 0,90: 0,3193
Сумма при x = 1,00: 0,1438
Сумма при x = 0,63: -0,7238
Сумма при x = 0,75: -0,6803
Сумма при x = 0,88: -0,6290
Сумма при x = 1,01: -0,5698
Сумма при x = 1,13: -0,5027
Сумма при x = 1,26: -0,4277
Сумма при x = 1,38: -0,3448
Сумма при x = 1,51: -0,2540
```

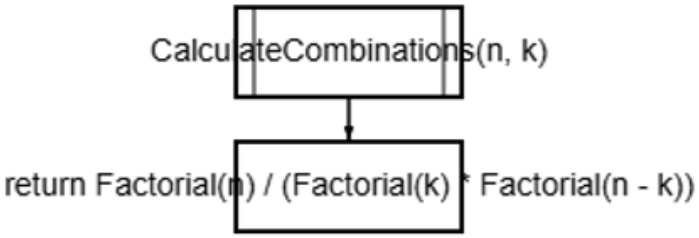

C:\Windows\system32\cmc
Сумма при x = 0,30: 1,4869
Сумма при x = 0,40: 1,3239
Сумма при x = 0,50: 1,1339
Сумма при x = 0,60: 0,9283
Сумма при x = 0,70: 0,7180
Сумма при x = 0,80: 0,5125
Сумма при x = 0,90: 0,3193
Сумма при x = 1,00: 0,1438
Сумма при x = 0,63: -0,7238
Сумма при x = 0,75: -0,6803
Сумма при x = 0,88: -0,6290
Сумма при x = 1,01: -0,5698
Сумма при x = 1,13: -0,5027
Сумма при x = 1,26: -0,4277
Сумма при x = 1,38: -0,3448
Сумма при x = 1,51: -0,2540
Сумма при x = 1,63: -0,1553
Сумма при x = 1,76: -0,0487
Сумма при x = 1,88: 0,0658
Сумма при x = 2,01: 0,1882
Сумма при x = 2,14: 0,3184
Сумма при x = 2,26: 0,4566
Сумма при x = 2,39: 0,6027
Сумма при x = 2,51: 0,7567
Сумма при x = 2,64: 0,9186
Сумма при x = 2,76: 1,0883
Сумма при x = 2,89: 1,2660
Сумма при x = 3,02: 1,4516
3 УРОВЕНЬ 2 ЗАДАЧА
Исходная матрица:
3 5 1 8
7 2 4 6
9 3 5 1
4 8 6 2
Матрица после обработки:
1 3 5 8
7 6 4 2
1 3 5 9
8 6 4 2
3 УРОВЕНЬ 3 ЗАДАЧА
Исходный массив:
1 3 5 7 9 11
Среднее арифметическое: 6
Массив после перестановки:
3 1 7 5 11 9
Сумма элементов с нечетными индексами: 15
Для продолжения нажмите любую клавишу . . .

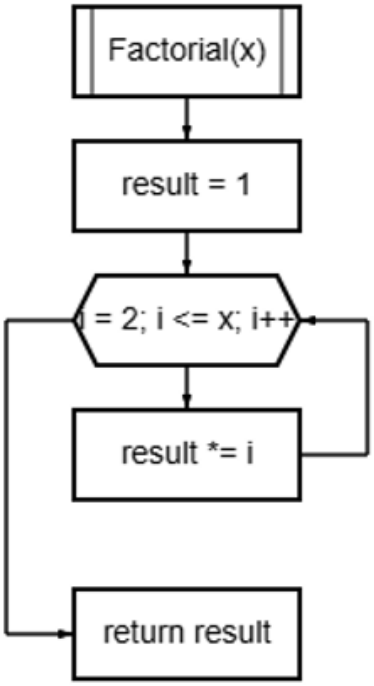
Блоксхемы для каждого задания:


Main()
1 УРОВЕНЬ 1 ЗАДАЧА
Task11.Run()
1 УРОВЕНЬ 2 ЗАДАЧА
Task12.Run()
1 УРОВЕНЬ 3 ЗАДАЧА
Task13.Run()
2 УРОВЕНЬ 25 ЗАДАЧА
Task225.Run()
2 УРОВЕНЬ 26 ЗАДАЧА
Task226.Run()
2 УРОВЕНЬ 27 ЗАДАЧА
Task227.Run()
3 УРОВЕНЬ 1 ЗАДАЧА
Task31.Run()
3 УРОВЕНЬ 2 ЗАДАЧА
Task32.Run()
3 УРОВЕНЬ 3 ЗАДАЧА
Task33.Run()

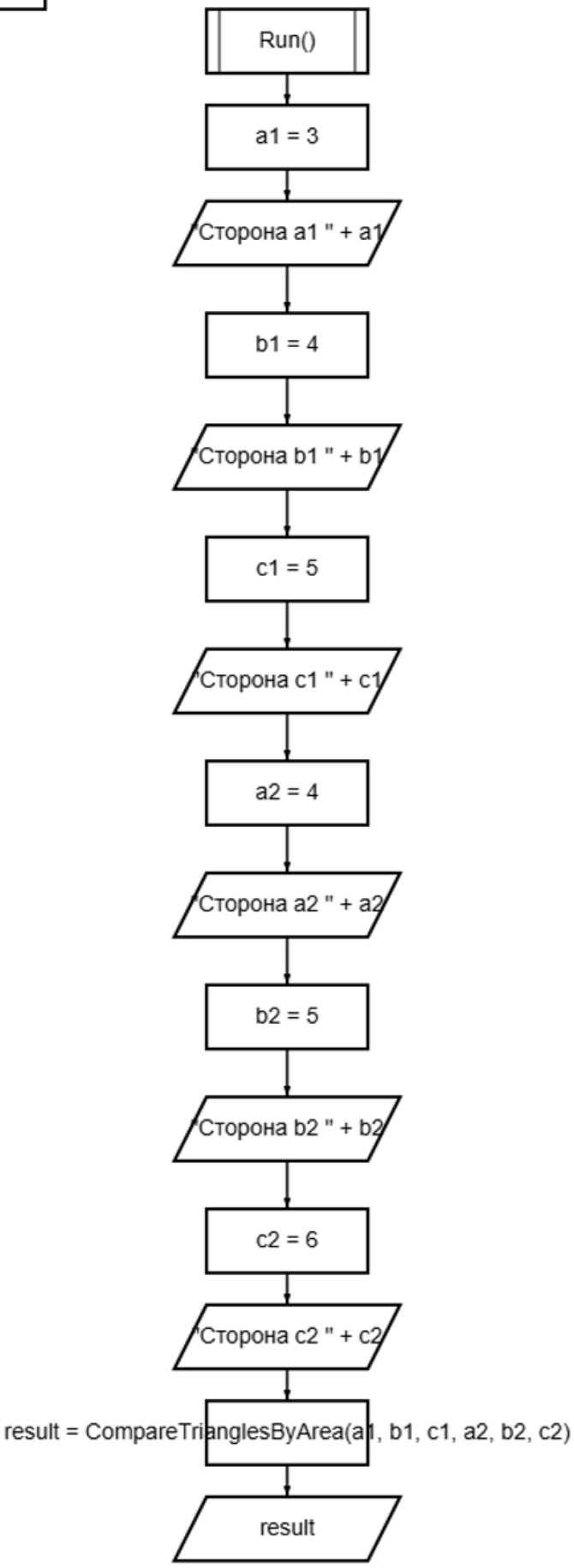
Task11

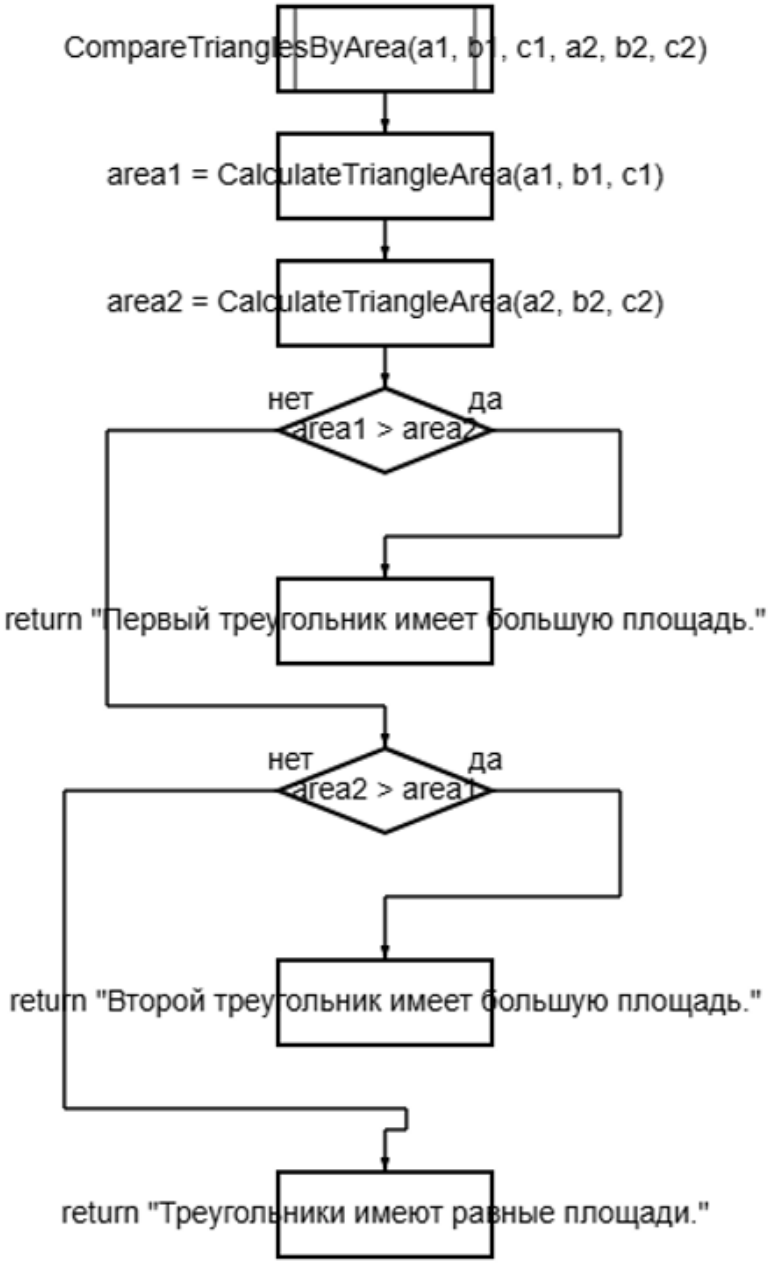

Run()
Количество способов отбора команды:
"Из 8 кандидатов: " + CalculateCombinations(8, 5)
"Из 10 кандидатов: " + CalculateCombinations(10, 5)
"Из 11 кандидатов: " + CalculateCombinations(11, 5)

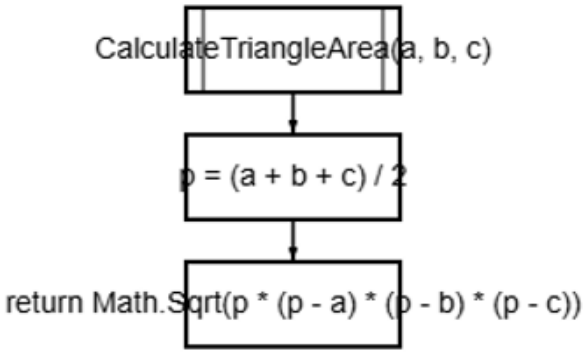

CalculateCombinations(n, k)
return Factorial(n) / (Factorial(k) * Factorial(n - k))


Factorial(x)
result = 1
i = 2; i <= x; i++
result *= i
return result

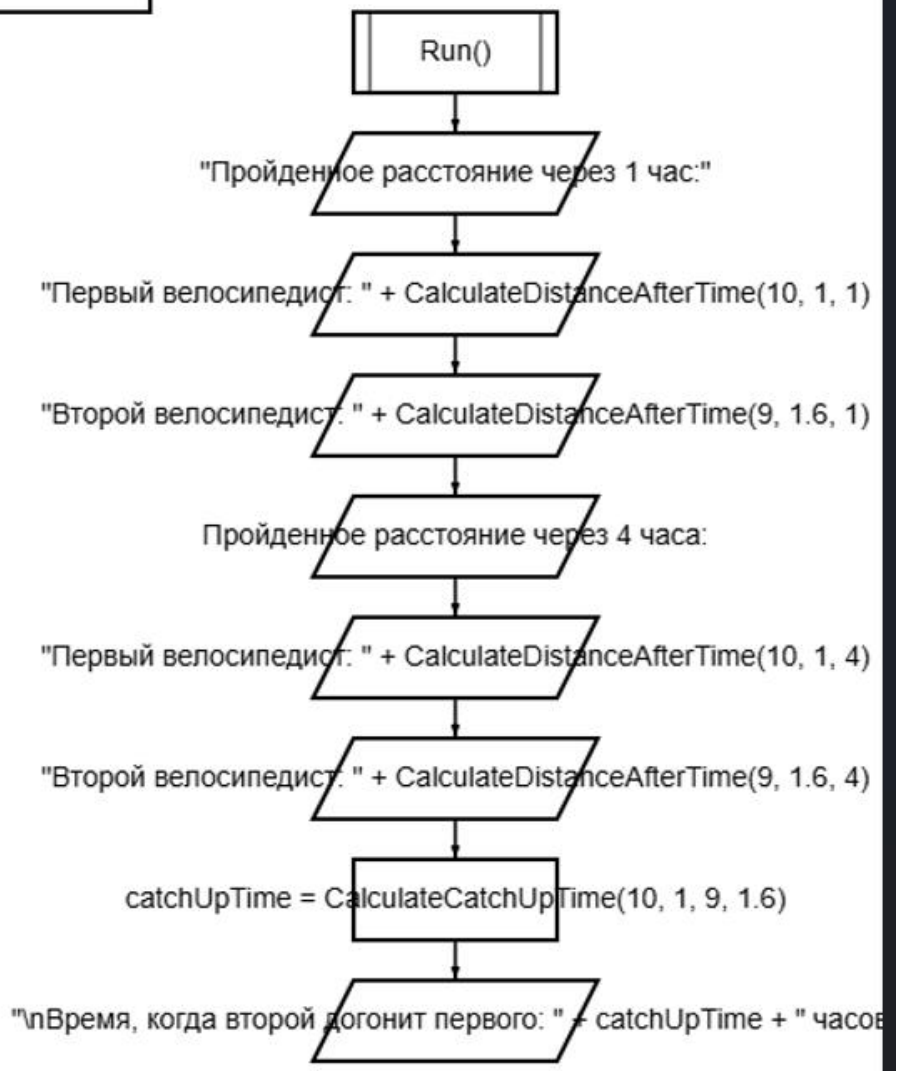
Task12

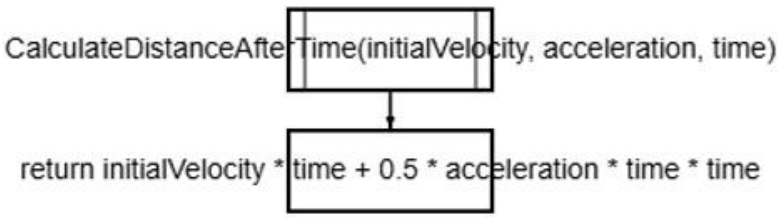

Run()
a1 = 3
"Сторона a1 " + a1
b1 = 4
"Сторона b1 " + b1
c1 = 5
"Сторона c1 " + c1
a2 = 4
"Сторона a2 " + a2
b2 = 5
"Сторона b2 " + b2
c2 = 6
"Сторона c2 " + c2
result = CompareTrianglesByArea(a1, b1, c1, a2, b2, c2)
result

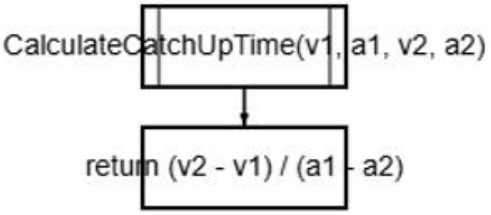

CompareTrianglesByArea(a1, b1, c1, a2, b2, c2)
area1 = CalculateTriangleArea(a1, b1, c1)
area2 = CalculateTriangleArea(a2, b2, c2)
нет
да
area1 > area2
return "Первый треугольник имеет большую площадь."
нет
да
area2 > area1
return "Второй треугольник имеет большую площадь."
return "Треугольники имеют равные площади."

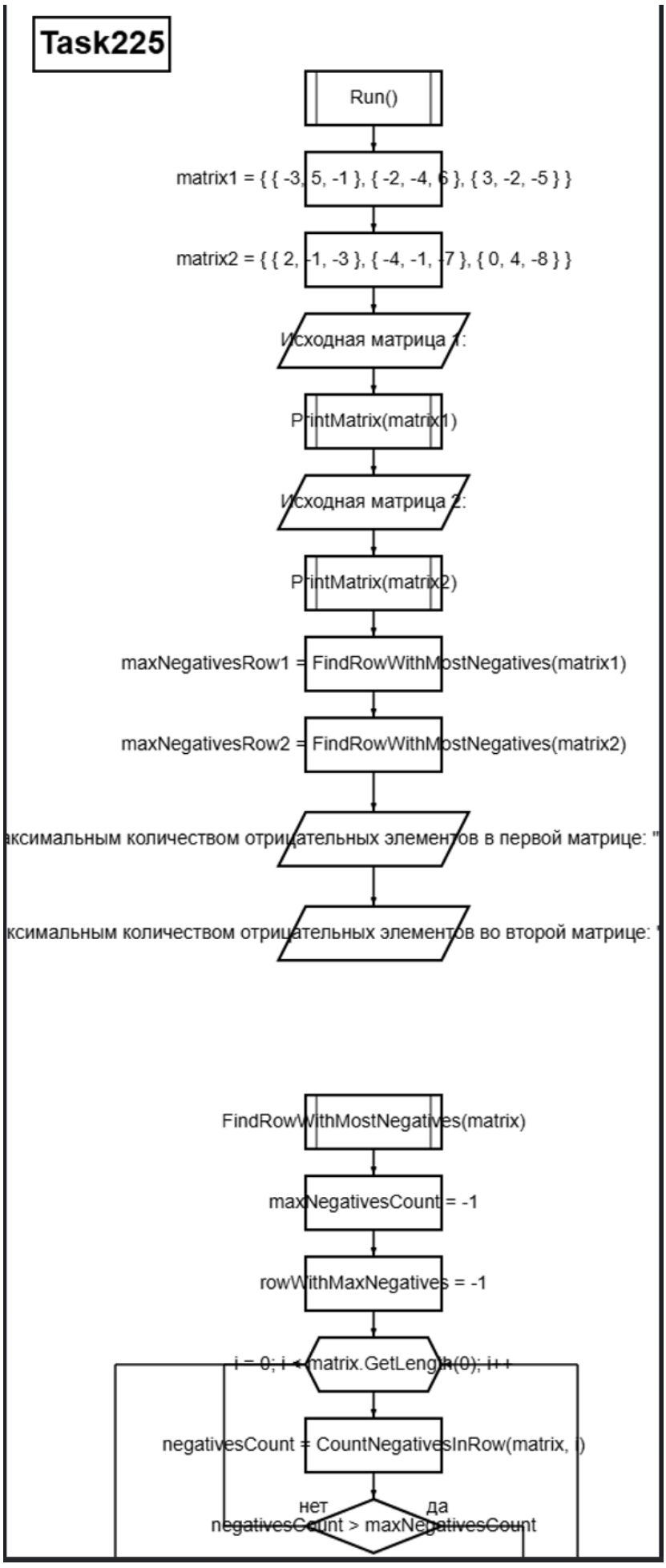

CalculateTriangleArea(a, b, c)
p = (a + b + c) / 2
return Math.Sqrt(p * (p - a) * (p - b) * (p - c))

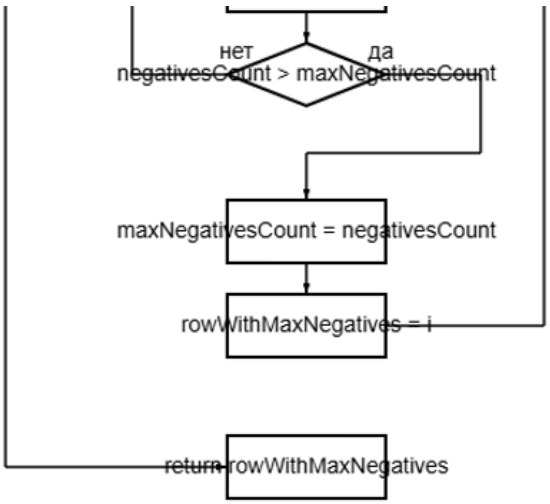
Task13

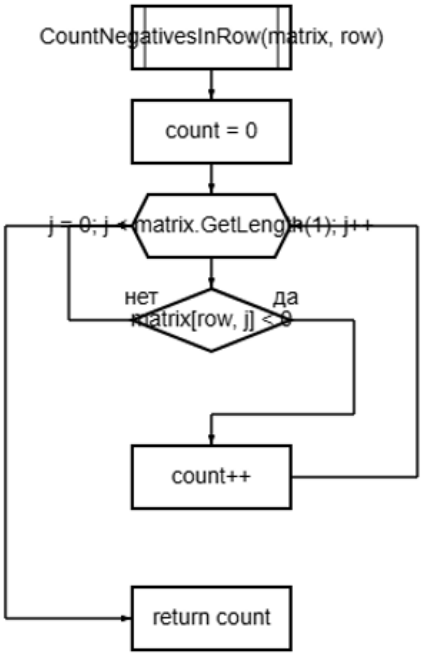

Run()
"Пройденное расстояние через 1 час:"
"Первый велосипедист: " + CalculateDistanceAfterTime(10, 1, 1)
"Второй велосипедист: " + CalculateDistanceAfterTime(9, 1.6, 1)
Пройденное расстояние через 4 часа:
"Первый велосипедист: " + CalculateDistanceAfterTime(10, 1, 4)
"Второй велосипедист: " + CalculateDistanceAfterTime(9, 1.6, 4)
catchUpTime = CalculateCatchUpTime(10, 1, 9, 1.6)
"\nВремя, когда второй догонит первого: " + catchUpTime + " часов

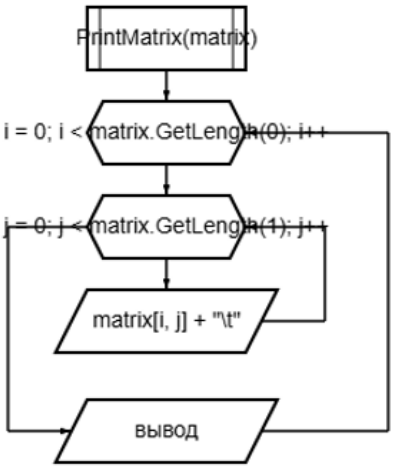

CalculateDistanceAfterTime(initialVelocity, acceleration, time)
return initialVelocity * time + 0.5 * acceleration * time * time


CalculateCatchUpTime(v1, a1, v2, a2)
return (v2 - v1) / (a1 - a2)

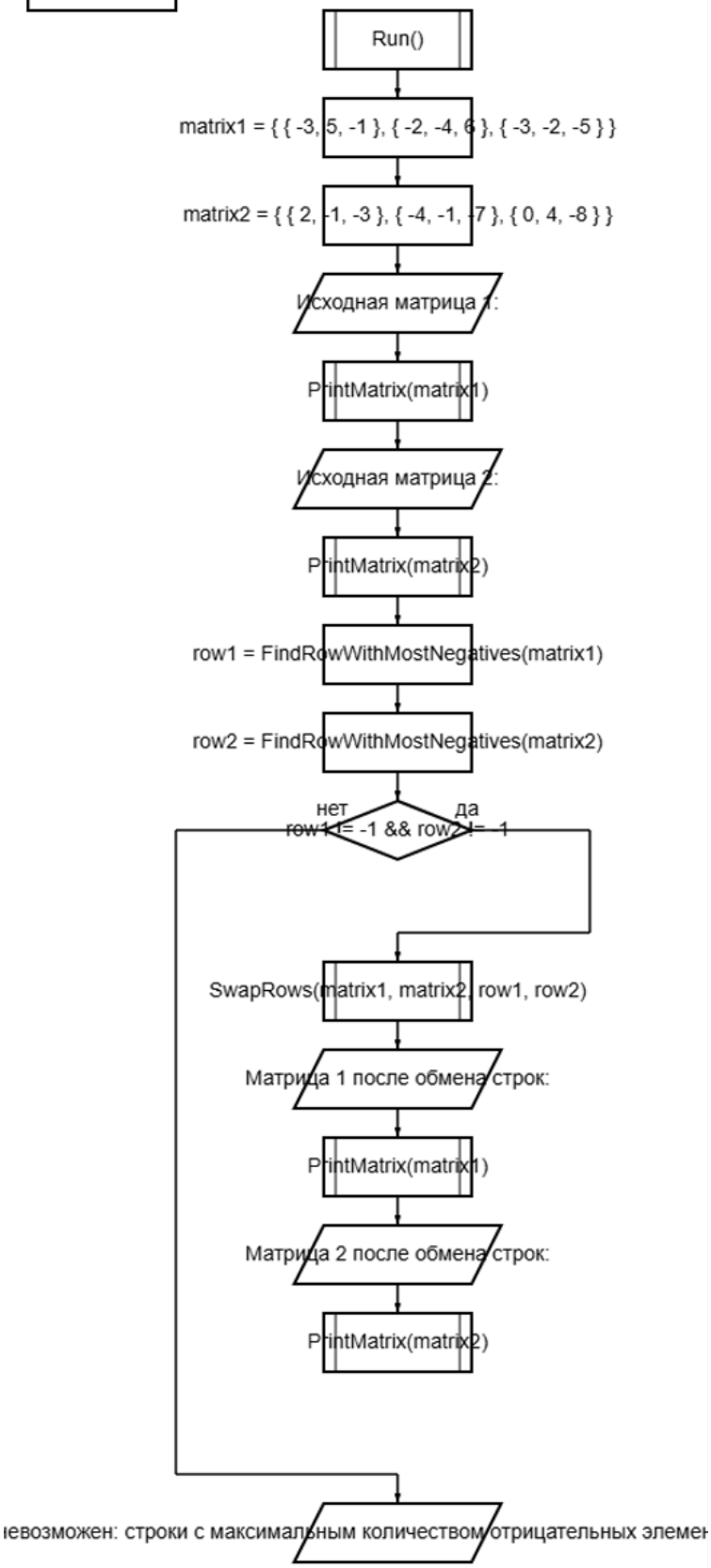

Task225
Run()
matrix1 = { { -3, 5, -1 }, { -2, -4, 6 }, { 3, -2, -5 } }
matrix2 = { { 2, -1, -3 }, { -4, -1, -7 }, { 0, 4, -8 } }
Исходная матрица 1:
PrintMatrix(matrix1)
Исходная матрица 2:
PrintMatrix(matrix2)
maxNegativesRow1 = FindRowWithMostNegatives(matrix1)
maxNegativesRow2 = FindRowWithMostNegatives(matrix2)
ксимальным количеством отрицательных элементов в первой матрице: "
ксимальным количеством отрицательных элементов во второй матрице: "
FindRowWithMostNegatives(matrix)
maxNegativesCount = -1
rowWithMaxNegatives = -1
i = 0; i < matrix.GetLength(0); i++
negativesCount = CountNegativesInRow(matrix, i)
нет
да
negativesCount > maxNegativesCount

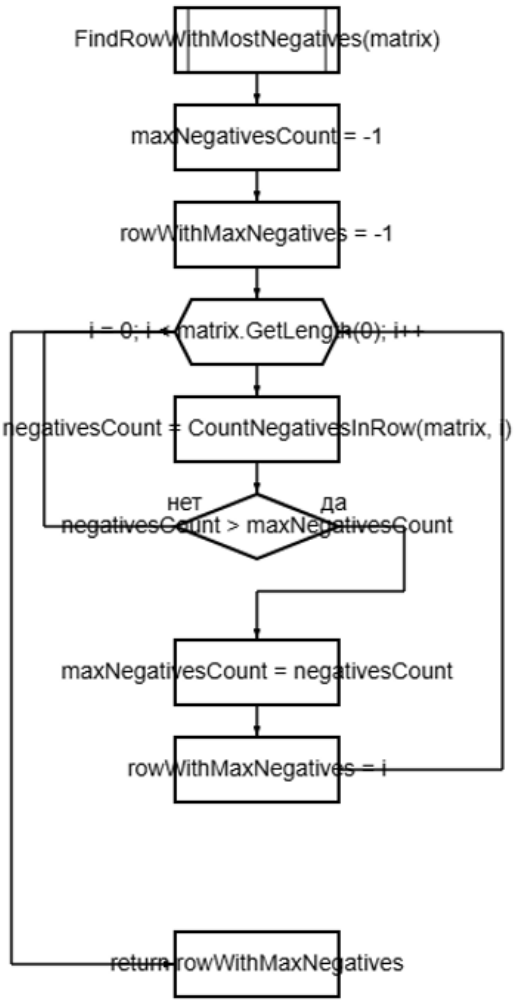

нет
да
negativesCount > maxNegativesCount
maxNegativesCount = negativesCount
rowWithMaxNegatives = i
return rowWithMaxNegatives

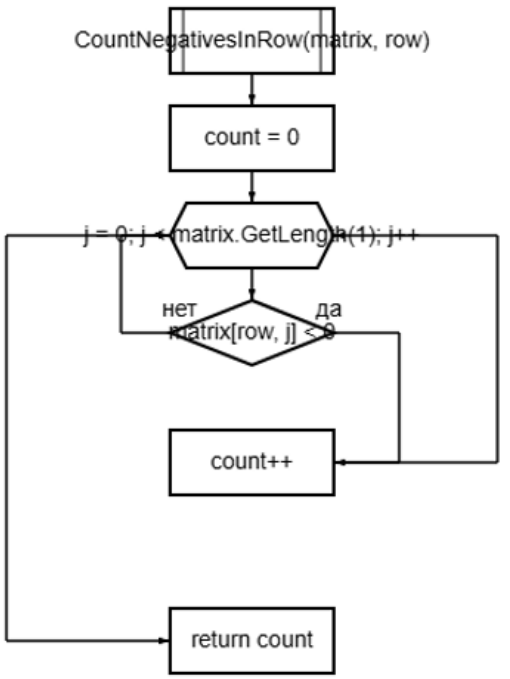

CountNegativesInRow(matrix, row)
count = 0
j = 0; j < matrix.GetLength(1); j++
нет
да
matrix[row, j] < 0
count++
return count

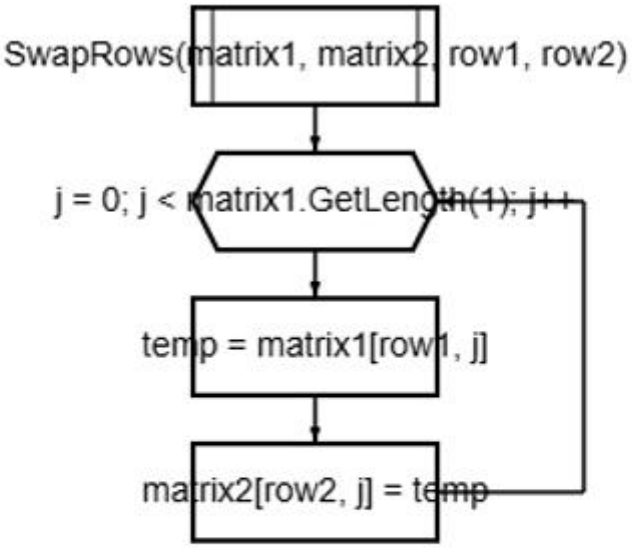

PrintMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
matrix[i, j] + "\t"
вывод

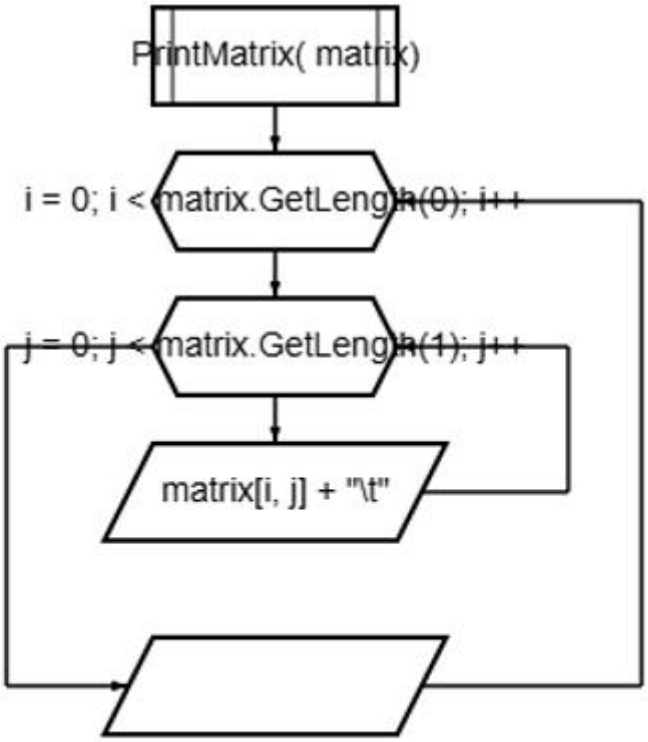
Task226


Run()
matrix1 = { { -3, 5, -1 }, { -2, -4, 6 }, { -3, -2, -5 } }
matrix2 = { { 2, -1, -3 }, { -4, -1, -7 }, { 0, 4, -8 } }
Исходная матрица 1:
PrintMatrix(matrix1)
Исходная матрица 2:
PrintMatrix(matrix2)
row1 = FindRowWithMostNegatives(matrix1)
row2 = FindRowWithMostNegatives(matrix2)
нет
да
row1 != -1 && row2 != -1
SwapRows(matrix1, matrix2, row1, row2)
Матрица 1 после обмена строк:
PrintMatrix(matrix1)
Матрица 2 после обмена строк:
PrintMatrix(matrix2)
невозможен: строки с максимальным количеством отрицательных элемен
FindRowWithMostNegatives(matrix)


FindRowWithMostNegatives(matrix)
maxNegativesCount = -1
rowWithMaxNegatives = -1
i = 0; i < matrix.GetLength(0); i++
negativesCount = CountNegativesInRow(matrix, i)
нет
да
negativesCount > maxNegativesCount
maxNegativesCount = negativesCount
rowWithMaxNegatives = i
return rowWithMaxNegatives

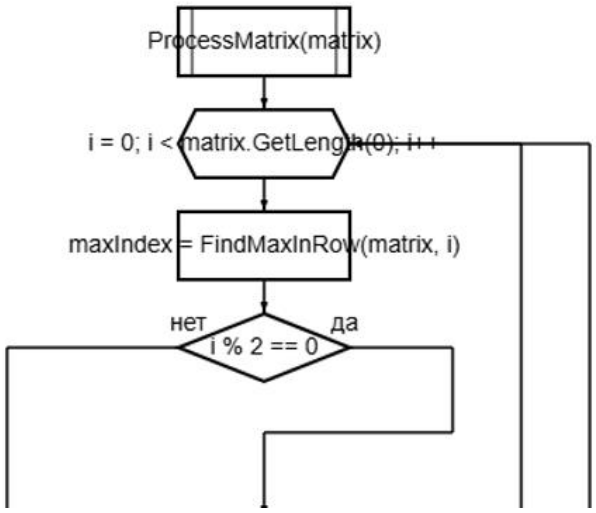

CountNegativesInRow(matrix, row)
count = 0
j = 0; j < matrix.GetLength(1); j++
нет
да
matrix[row, j] < 0
count++
return count

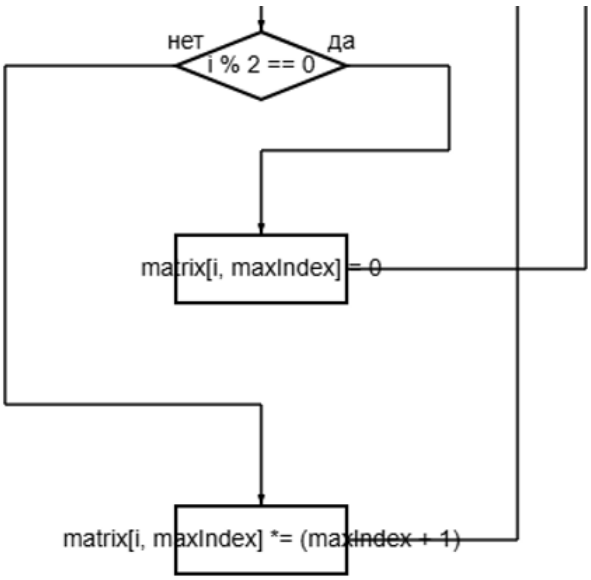

SwapRows(matrix1, matrix2, row1, row2)
j = 0; j < matrix1.GetLength(1); j++
temp = matrix1[row1, j]
matrix2[row2, j] = temp

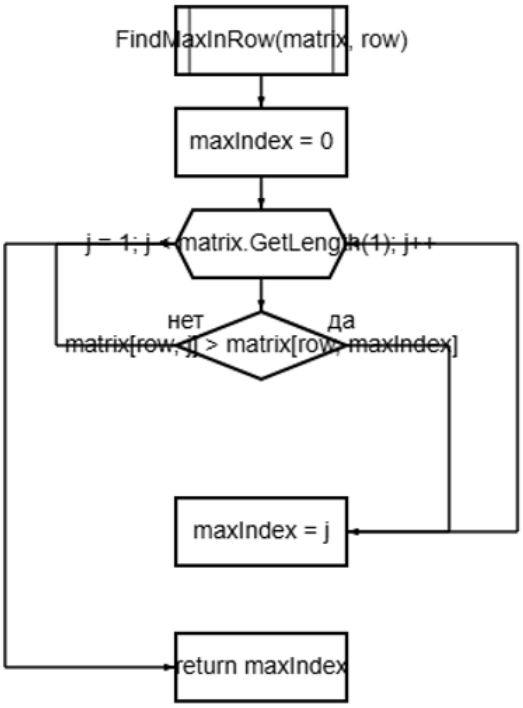

PrintMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
matrix[i, j] + "\t"

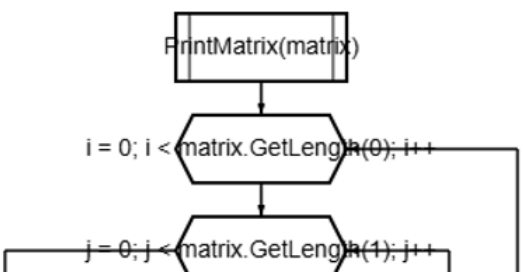
Task227

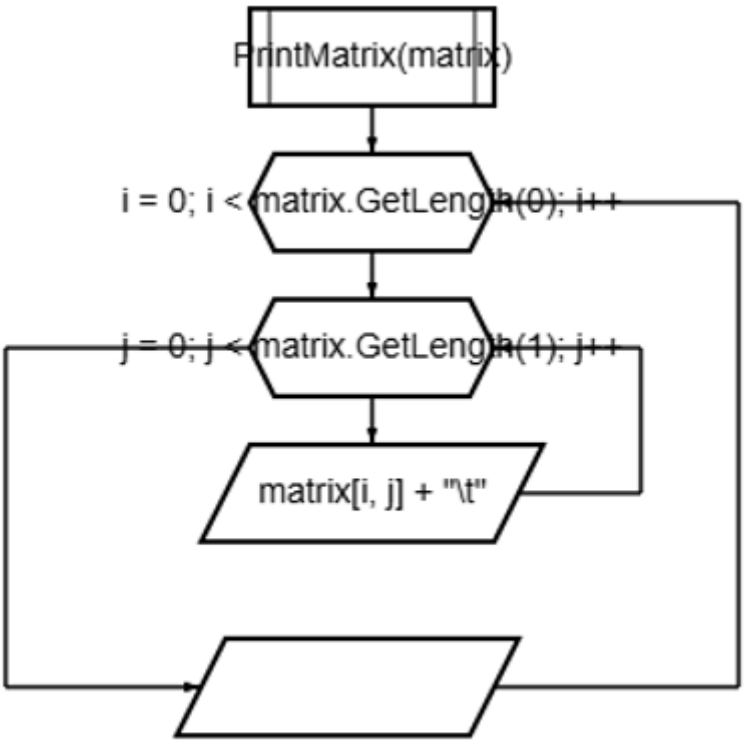

Run()
matrix1 = { { 2, -1, 4 }, { 3, -4, -7 }, { 5, -2, 8 } }
matrix2 = { { -6, 3, -2 }, { 8, 1, -5 }, { 7, -3, 2 } }
Исходная матрица 1:
PrintMatrix(matrix1)
Исходная матрица 2:
PrintMatrix(matrix2)
Матрица 1 после обработки:
PrintMatrix(matrix1)
Матрица 2 после обработки:
PrintMatrix(matrix2)


ProcessMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
maxIndex = FindMaxInRow(matrix, i)
нет
да
i % 2 == 0

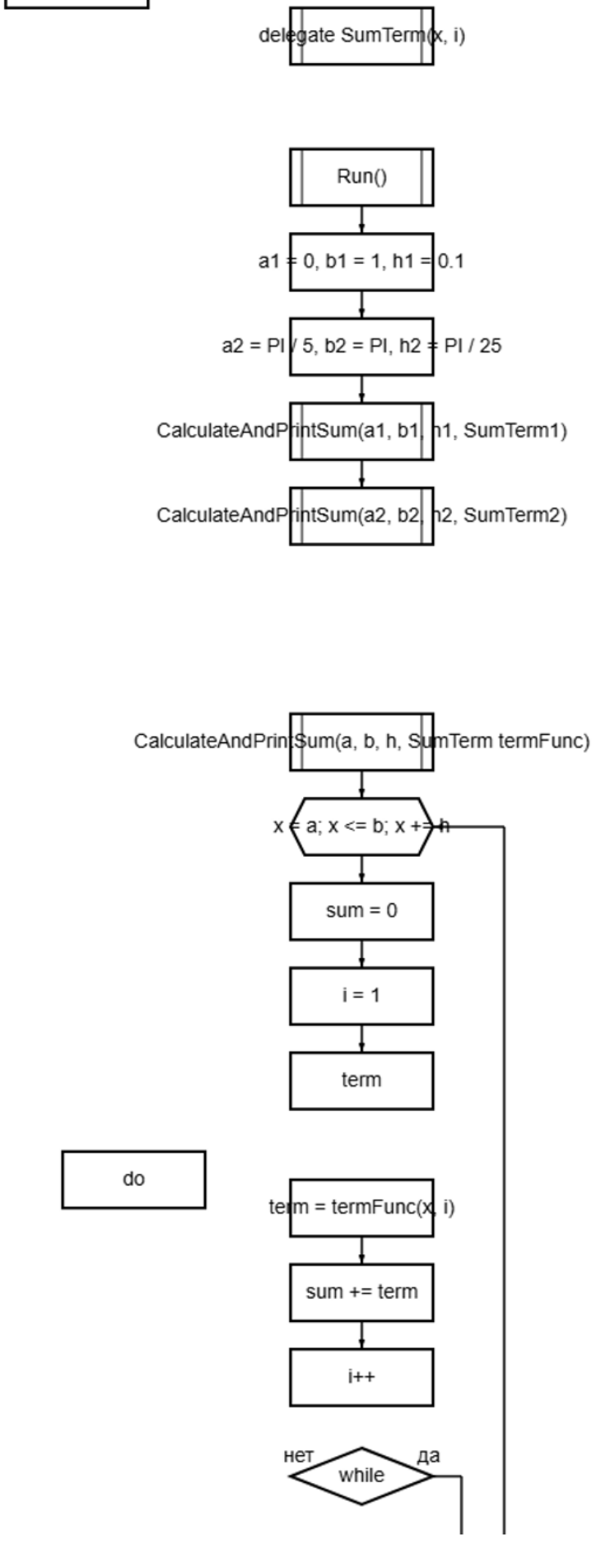

нет
i % 2 == 0
да
matrix[i, maxIndex] = 0
matrix[i, maxIndex] *= (maxIndex + 1)

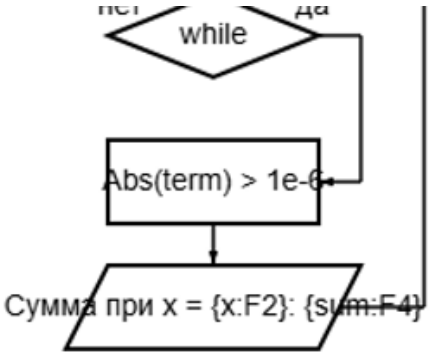

FindMaxInRow(matrix, row)
maxIndex = 0
j = 1; j < matrix.GetLength(1); j++
нет
да
matrix[row, j] > matrix[row, maxIndex]
maxIndex = j
return maxIndex

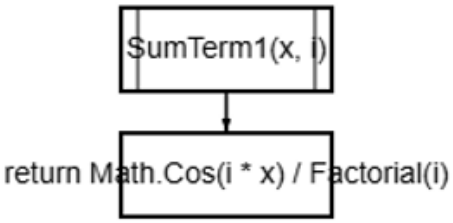

PrintMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++

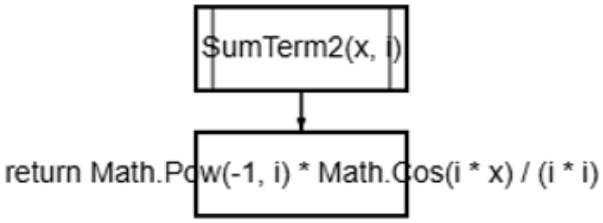

PrintMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
matrix[i, j] + "\t"

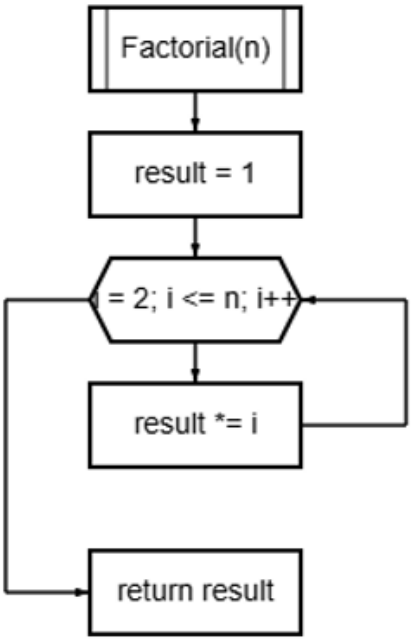
Task31

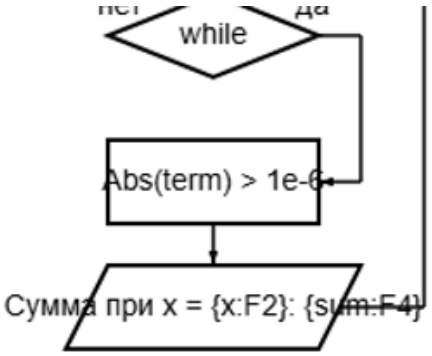

delegate SumTerm(x, i)
Run()
a1 = 0, b1 = 1, h1 = 0.1
a2 = PI / 5, b2 = PI, h2 = PI / 25
CalculateAndPrintSum(a1, b1, h1, SumTerm1)
CalculateAndPrintSum(a2, b2, h2, SumTerm2)
CalculateAndPrintSum(a, b, h, SumTerm termFunc)
x = a; x <= b; x += h
sum = 0
i = 1
term
do
term = termFunc(x, i)
sum += term
i++
нет
while
да

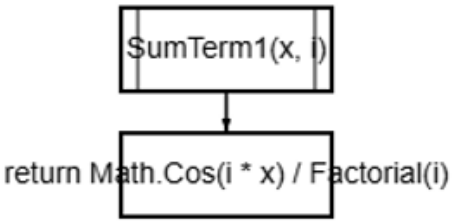

while
Abs(term) > 1e-6
Сумма при x = {x:F2}: {sum:F4}

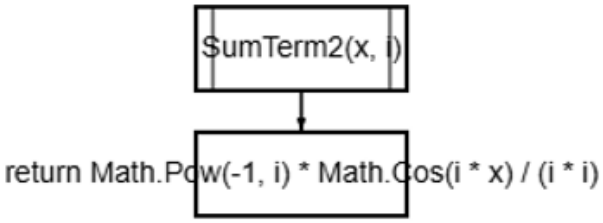

SumTerm1(x, i)
return Math.Cos(i * x) / Factorial(i)

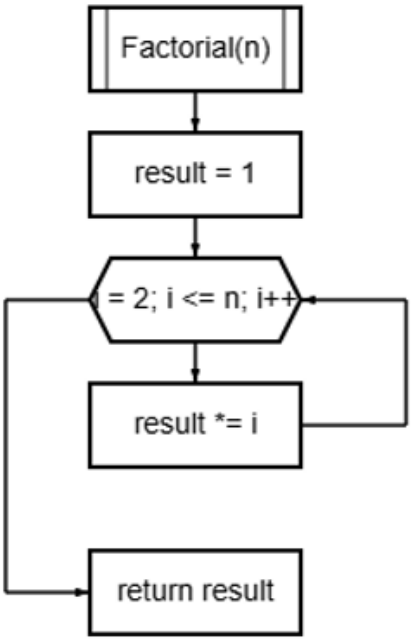

SumTerm2(x, i)
return Math.Pow(-1, i) * Math.Cos(i * x) / (i * i)

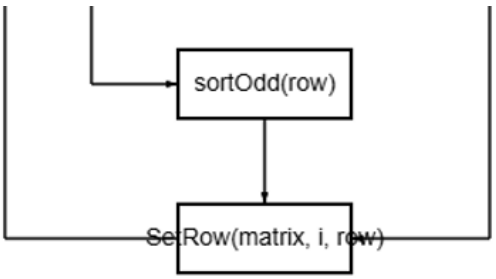

Factorial(n)
result = 1
= 2; i <= n; i++
result *= i
return result

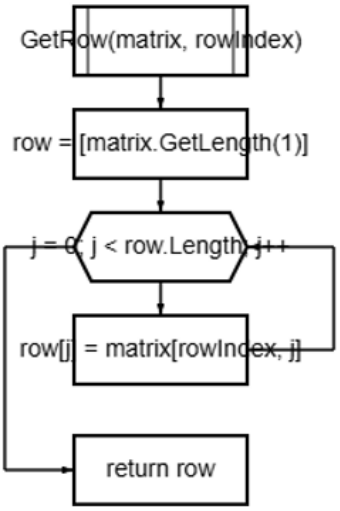

while
Abs(term) > 1e-6
Сумма при x = {x:F2}: {sum:F4}

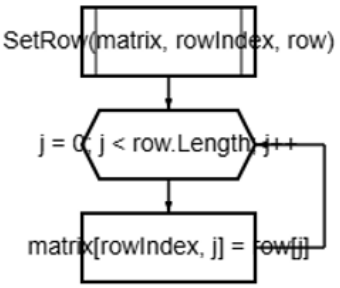

SumTerm1(x, i)
return Math.Cos(i * x) / Factorial(i)

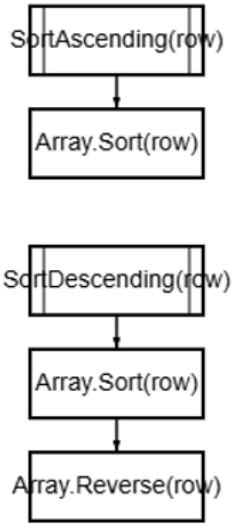

SumTerm2(x, i)
return Math.Pow(-1, i) * Math.Cos(i * x) / (i * i)

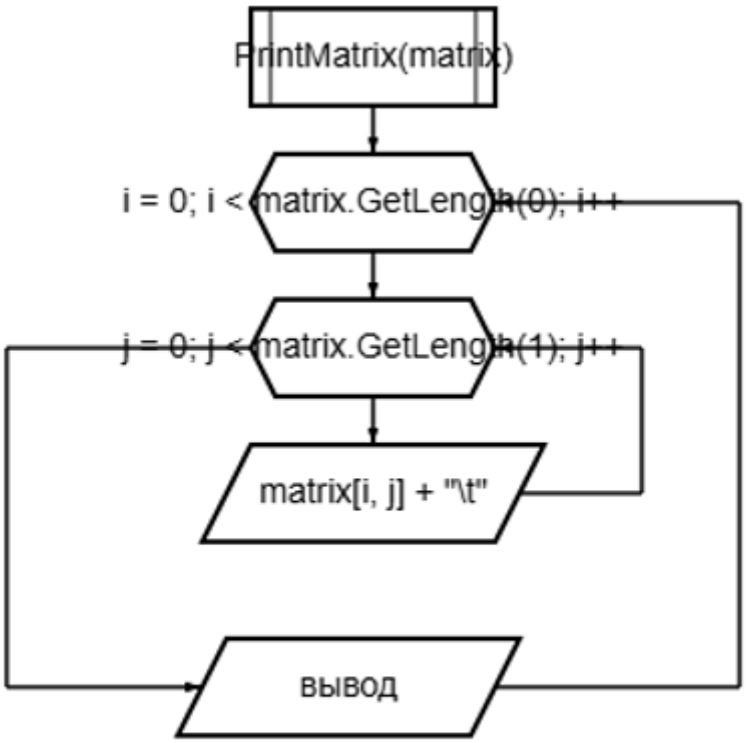

Factorial(n)
result = 1
= 2; i <= n; i++
result *= i
return result


sortOdd(row)
SetRow(matrix, i, row)

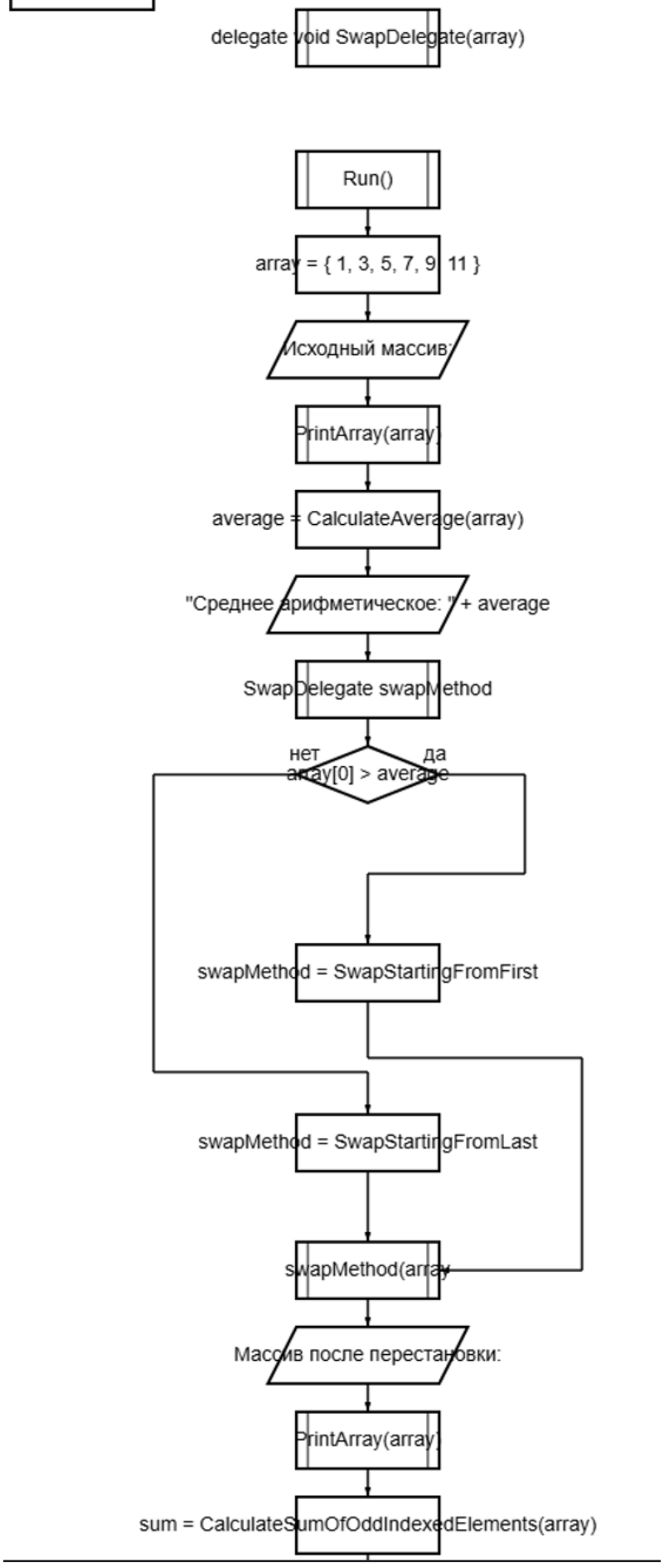

GetRow(matrix, rowIndex)
row = [matrix.GetLength(1)]
j = 0; j < row.Length; j++
row[j] = matrix[rowIndex, j]
return row


SetRow(matrix, rowIndex, row)
j = 0; j < row.Length; j++
matrix[rowIndex, j] = row[j]

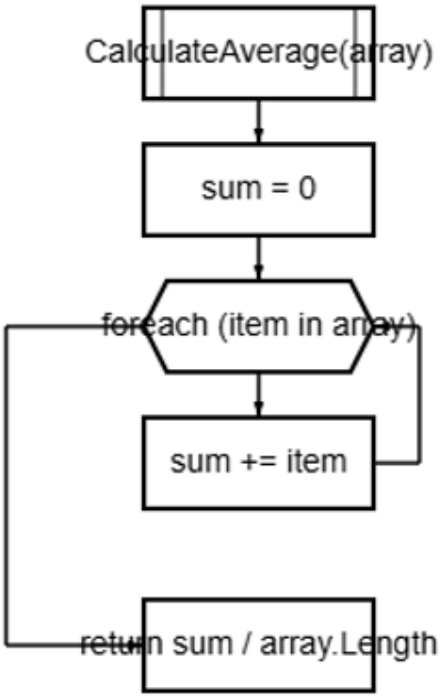

SortAscending(row)
Array.Sort(row)
SortDescending(row)
Array.Sort(row)
Array.Reverse(row)

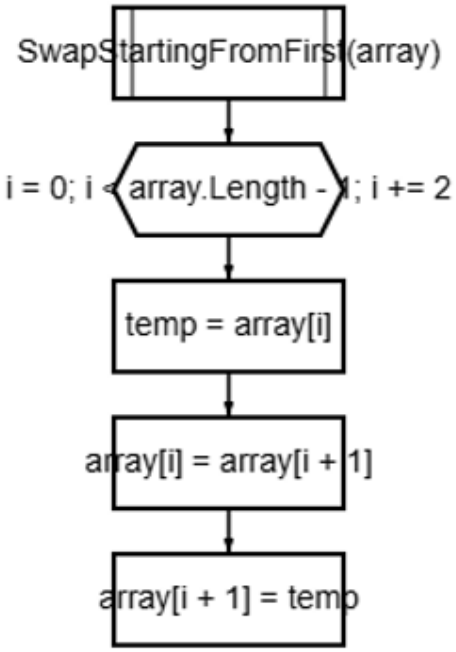

PrintMatrix(matrix)
i = 0; i < matrix.GetLength(0); i++
j = 0; j < matrix.GetLength(1); j++
matrix[i, j] + "\t"
вывод

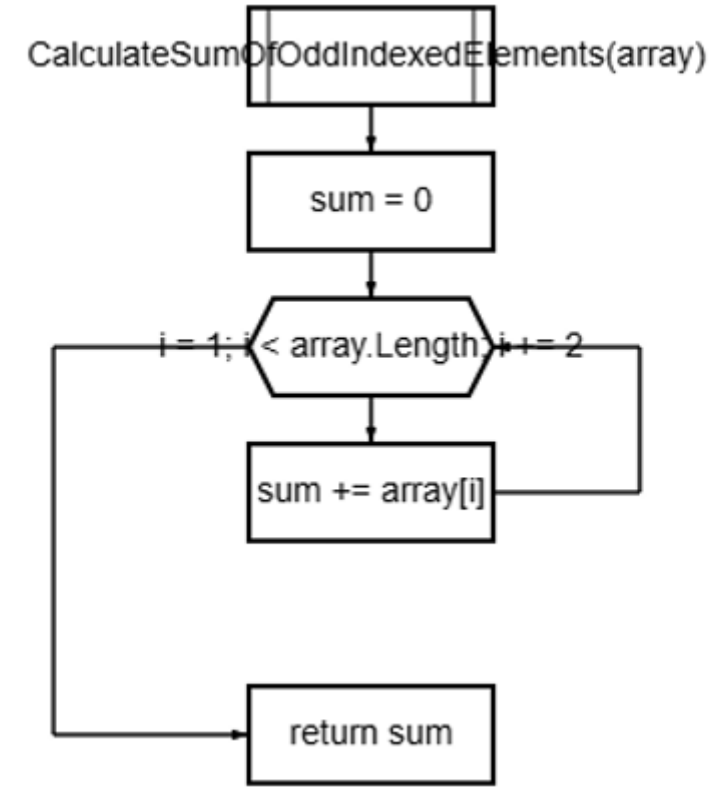
Task33

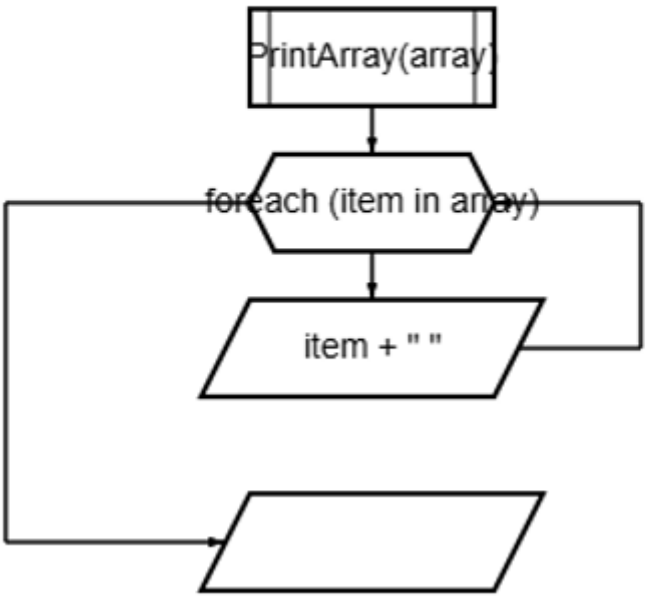

delegate void SwapDelegate(array)
Run()
array = { 1, 3, 5, 7, 9, 11 }
Исходный массив:
PrintArray(array)
average = CalculateAverage(array)
"Среднее арифметическое: " + average
SwapDelegate swapMethod
нет
да
array[0] > average
swapMethod = SwapStartingFromFirst
swapMethod = SwapStartingFromLast
swapMethod(array
Массив после перестановки:
PrintArray(array)
sum = CalculateSumOfOddIndexedElements(array)


sum = CalculateSumOfOddIndexedElements(array)
"Сумма элементов с нечетными индексами: " + sum


CalculateAverage(array)
sum = 0
foreach (item in array)
sum += item
return sum / array.Length


SwapStartingFromFirst(array)
i = 0; i < array.Length - 1; i += 2
temp = array[i]
array[i] = array[i + 1]
array[i + 1] = temp





Вывод: в ходе выполнения лабораторной работы были приобретены практические навыки создания и использования методов в языке C#. Мы изучили, как с помощью методов структурировать программу, облегчая её восприятие и упрощая отладку. Были разработаны методы для выполнения различных вычислений, обработки массивов и работы с матрицами, а также реализованы задачи с использованием делегатов, что позволило повысить

гибкость и универсальность кода. Выполнение работы позволило лучше понять принципы объектно-ориентированного программирования, где методы являются неотъемлемой частью работы с объектами, и научило применять делегаты для передачи функций в качестве параметров методов.