**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«РОССИЙСКАЯ АКАДЕМИЯ НАРОДНОГО ХОЗЯЙСТВА   
И ГОСУДАРСТВЕННОЙ СЛУЖБЫ**

**ПРИ ПРЕЗИДЕНТЕ РОССИЙСКОЙ ФЕДЕРАЦИИ»**

**НИЖЕГОРОДСКИЙ ИНСТИТУТ УПРАВЛЕНИЯ – филиал РАНХиГС**

Факультет управления

Кафедра Информатики и информационных технологий

Направление подготовки / специальность: 09.04.03 Прикладная информатика

Направленность (профиль) / специализация: Информационные технологии в ГМУ или Корпоративные информационные системы управления или ещё как-то

**Отчет по лабораторным работам**

|  |  |
| --- | --- |
| по дисциплине: | Структуры данных и их использование в программировании |
|  |  |
|  |  |
|  |  |

|  |  |  |
| --- | --- | --- |
|  |  | **АВТОР** |
|  |  | Обучающийся(иеся) 2 курса группы Иб-321 |
|  |  | очной формы обучения |
|  |  |  |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ Комышков Владислав Дмитриевич |
|  |  | *(подпись) (фамилия, инициалы)* |
|  |  |  |
|  |  | **РУКОВОДИТЕЛЬ** |
|  |  | Кандидат физико-математических наук |
|  |  | *(ученая степень, ученое звание)* |
|  |  | оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  | «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г. |
|  |  | *(дата защиты)* |
|  |  | \_\_\_\_\_\_\_\_\_\_\_\_\_ Орлов М.Л. |
|  |  | *(подпись) (фамилия, инициалы)* |

**Оглавление**

[Практическое задание 1 4](#_Toc122190354)

[Задание 1 4](#_Toc122190355)

[Program.cs 6](#_Toc122190356)

[Задание 2 9](#_Toc122190357)

[Program.cs 11](#_Toc122190358)

[Задание 3 14](#_Toc122190359)

[Program.cs 19](#_Toc122190360)

[Практическое задание 2 24](#_Toc122190361)

[Задание 1 24](#_Toc122190362)

[Program.cs 24](#_Toc122190363)

[Задание 2 26](#_Toc122190364)

[Program.cs 26](#_Toc122190365)

[Задание 3 30](#_Toc122190366)

[Program.cs 30](#_Toc122190367)

[Задание 4 34](#_Toc122190368)

[Program.cs 34](#_Toc122190369)

[Задание 5 39](#_Toc122190370)

[Program.cs 39](#_Toc122190371)

[Задание 6 45](#_Toc122190372)

[Program.cs 46](#_Toc122190373)

[Задание 7 54](#_Toc122190374)

[Program.cs 54](#_Toc122190375)

[Практическое задание 3 58](#_Toc122190376)

[Задание 1 58](#_Toc122190377)

[Program.cs 58](#_Toc122190378)

[Form1.cs 59](#_Toc122190379)

[MethodsSort.cs 66](#_Toc122190380)

[Задание 2 90](#_Toc122190381)

[Form2.cs 90](#_Toc122190382)

[Практическое задание 4 111](#_Toc122190383)

[Задание 1 111](#_Toc122190384)

[Program.cs 111](#_Toc122190385)

[Задание 2 116](#_Toc122190386)

[Program.cs 116](#_Toc122190387)

[Практическое задание 5 120](#_Toc122190388)

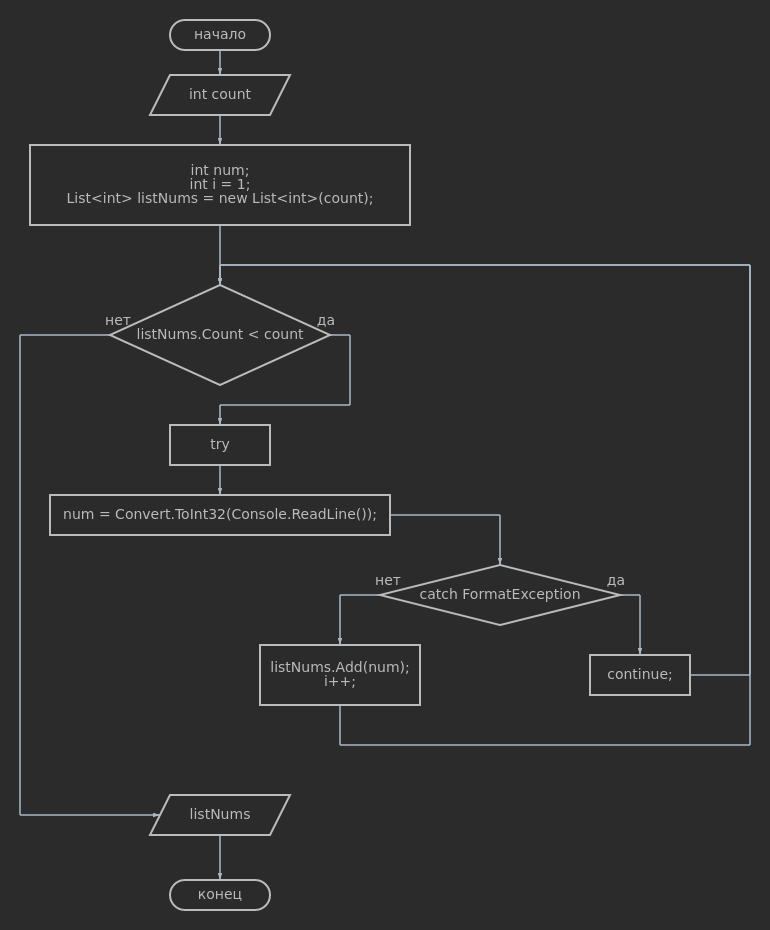
[Задание 1 120](#_Toc122190389)

[Program.cs 120](#_Toc122190390)

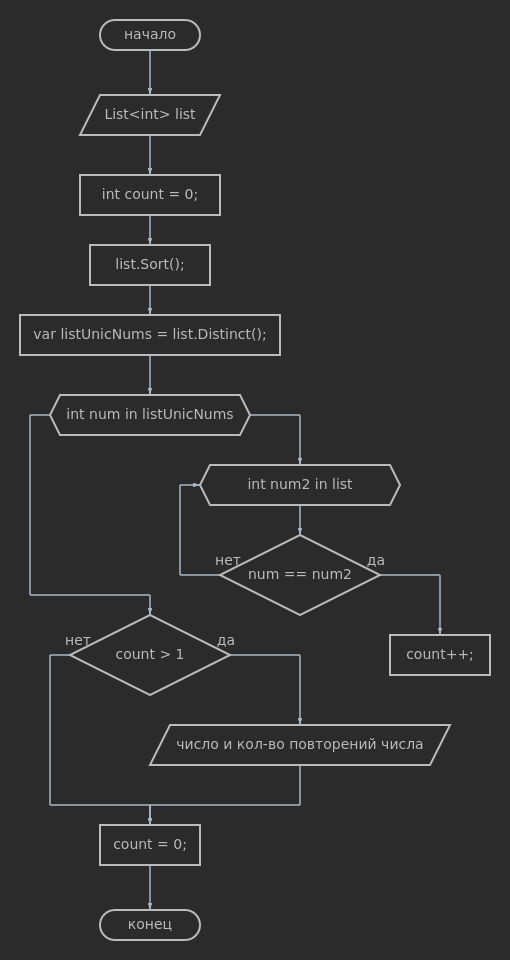
# Практическое задание 1

## Задание 1

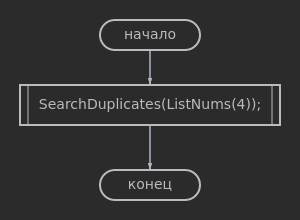
Разработайте алгоритм, который запрашивает у пользователя четыре целых числа и затем определяет, есть ли среди введенных чисел одинаковые или нет. Нарисуйте блок-схему алгоритма. Программный код может быть реализован на одном из следующих языков программирования: С, С++, С#, Java.

****Блок-схема**

*diagram-ListNums*

**

*SearchDuplicates*

**

*diagram-Main*

### Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Task1

{

    class Program

    {

*public* *static* List<int> ListNums(int count) *// запрашивает у пользователя желаемое кол-во чисел и возвращет список введенных чисел*

        {

            int num; *// переменная в которой будут храниться по очередно вводимые числа*

            int i = 1; *// номер вводимого числа*

            List<int> listNums = new List<int>(count); *// список в котором будут храниться числа*

            while (listNums.Count < count) *// пока длина списка не равна желаемому кол-ву чисел*

            {

                try *// пытаемся привести введенное число к типу данных int*

                {

                    Console.Write("Введите {0}-е число: ", i);

                    num = Convert.ToInt32(Console.ReadLine()); *// присваеваем переменной введенное число и преобразуем в тип данных int*

                }

                catch (FormatException) *// если поймана ошибка при попытке преобразовать число*

                {

                    Console.WriteLine("Ожидается число...");

                    continue; *// цикл запускается заново*

                }

                listNums.Add(num); *// добавляем введенное число в список*

                i++; *// номер следующего числа*

            }

            return listNums; *// возращается список*

        }

*public* *static* void SearchDuplicates(List<int> list) *// поиск дубликатов в списке чисел*

        {

            int count = 0; *// переменная для хранения кол-ва дубликатов*

            list.Sort(); *// сортируем список*

            var listUnicNums = list.Distinct(); *// преобразуем список чисел в список неповторяющихся чисел (не индексируется)*

            foreach (int num in listUnicNums) *// перебираем список неповторяющихся чисел и каждое присваеваем переменной num*

            {

                foreach (int num2 in list) *// перебираем список чисел и каждое присваеваем переменной num2*

                {

                    if (num == num2) *// если числа из списков равны*

                        count++; *// увеличиваем кол-во дубликатов*

                }

                if (count > 1) *// если кол-во дубликатов числа больше 1*

                {

                    Console.WriteLine("Число {0} повторяется {1} раз(а)", num, count); *// сообщаем пользователю информацию о дубликатах*

                }

                count = 0; *// обнуляем кол-во дубликатов, чтобы посчитать дубликаты следующих чисел в списке*

            }

        }

*public* *static* void Main(string[] args)

        {

*/\**

*\* запрашиваем у пользователя желаемое кол-во чисел и возвращем список введенных чисел*

*\* ищем дубликаты чисел в списке*

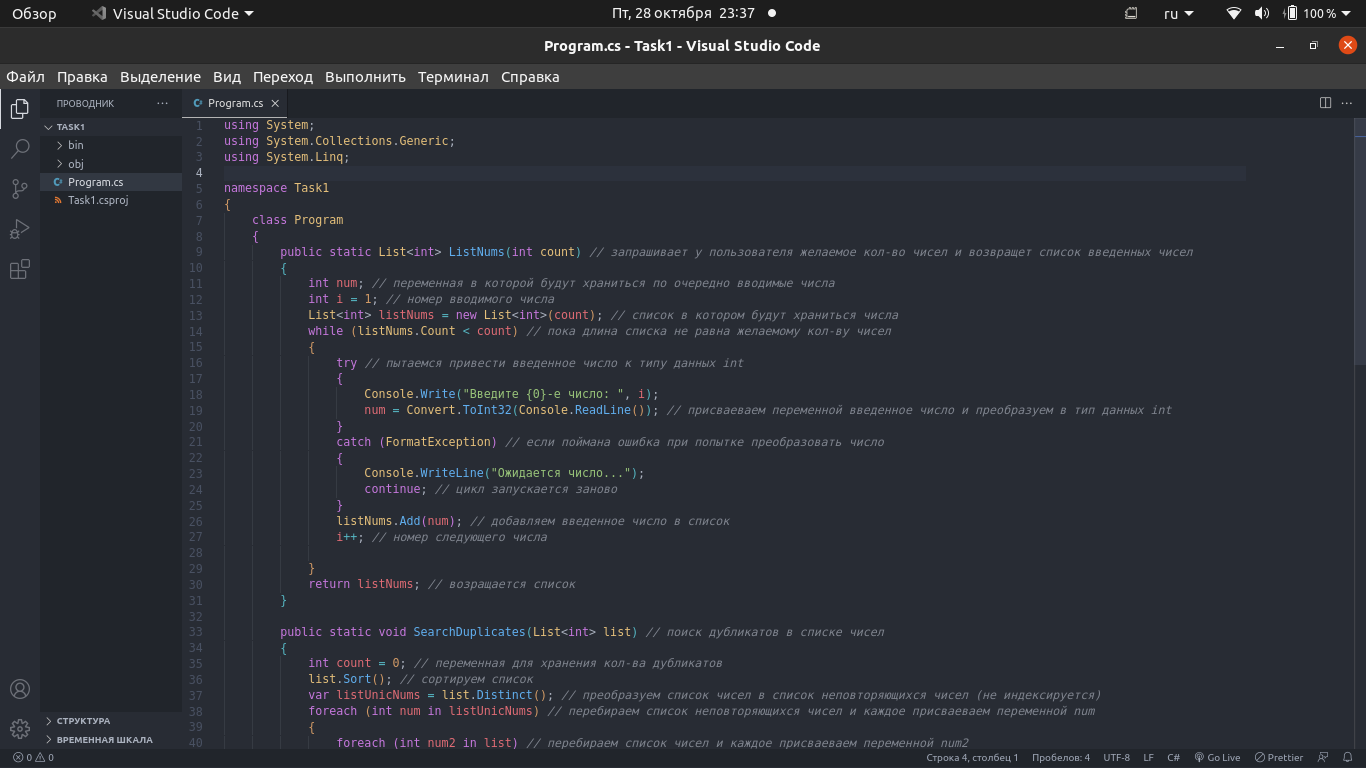
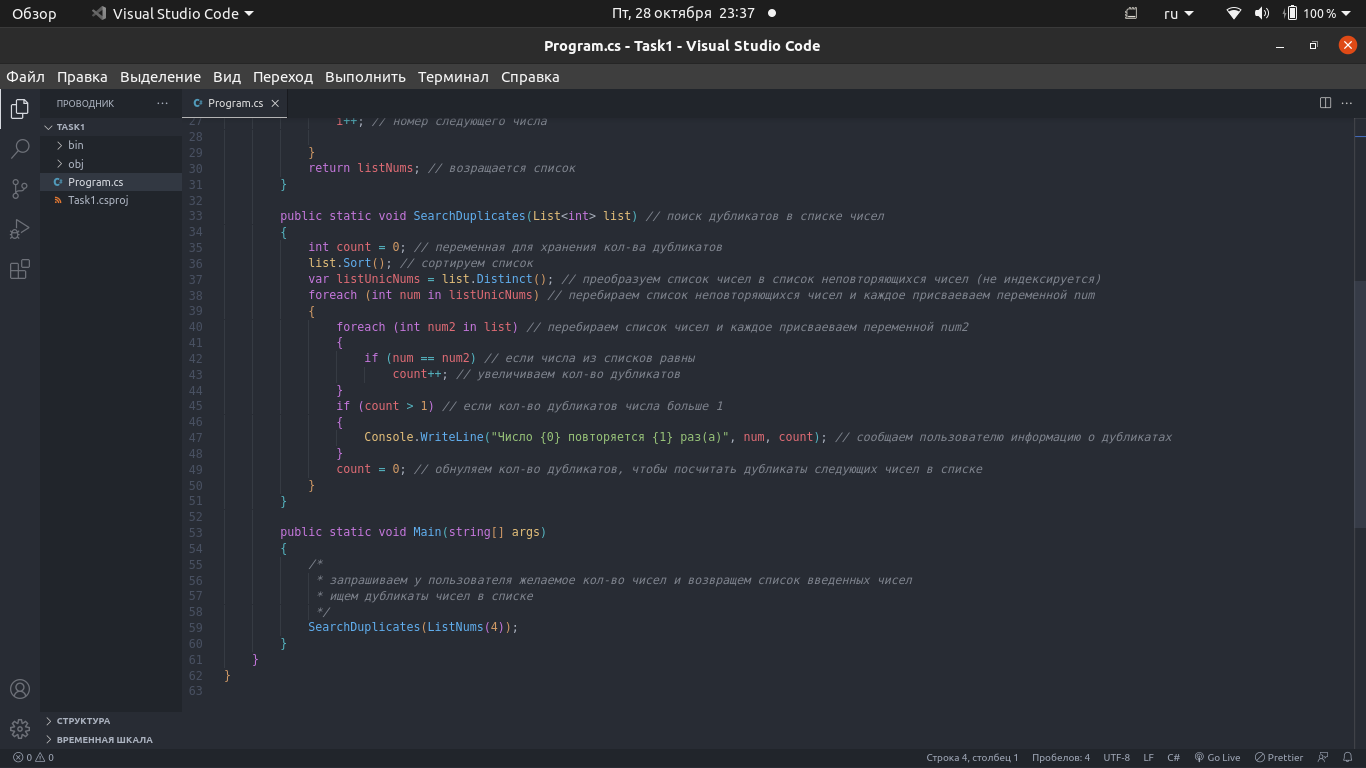
*\*/*

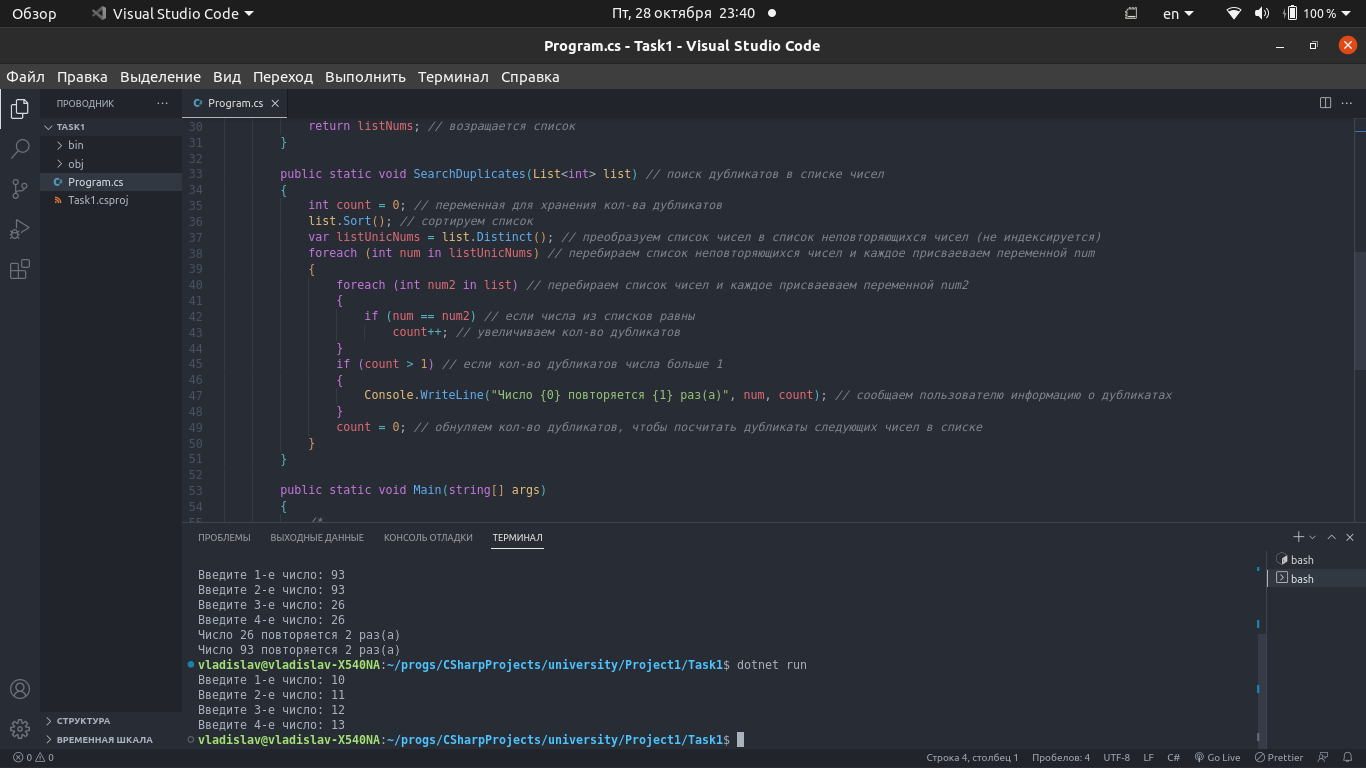
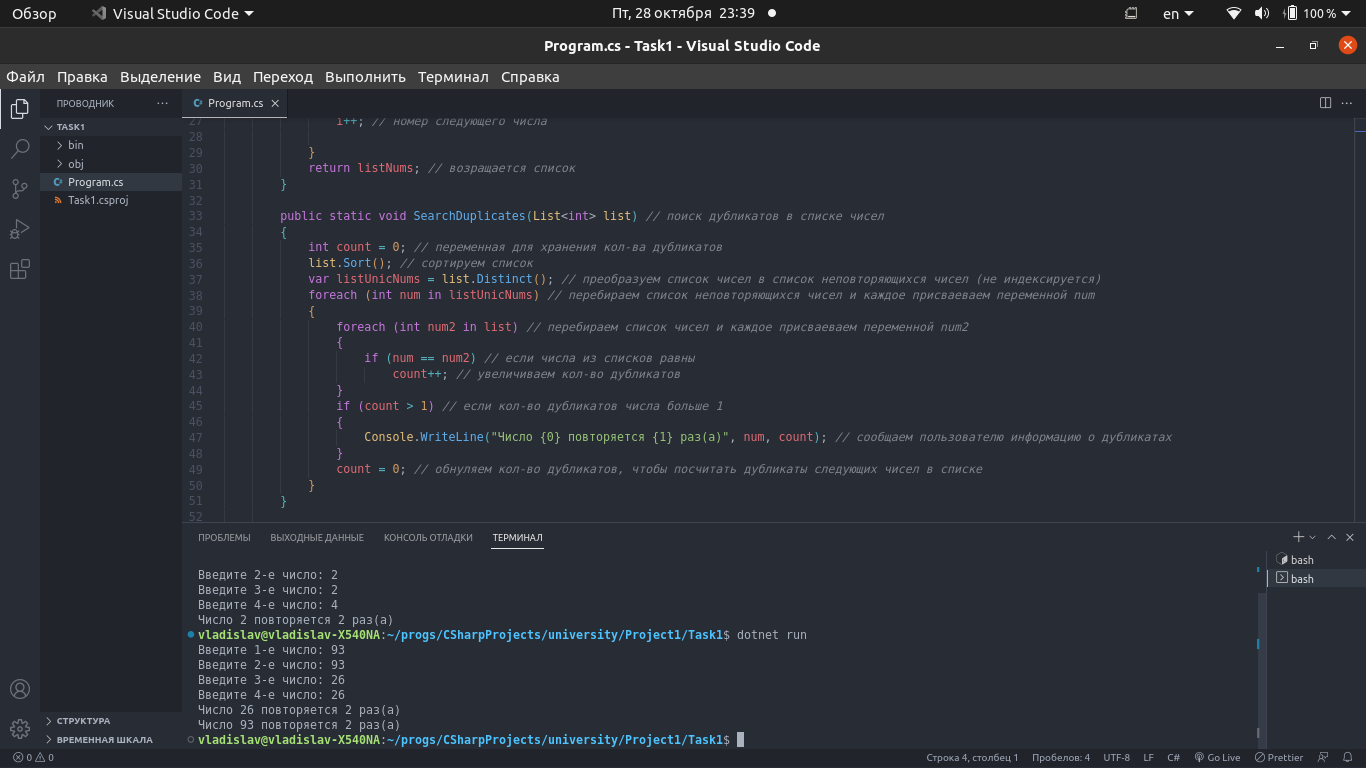
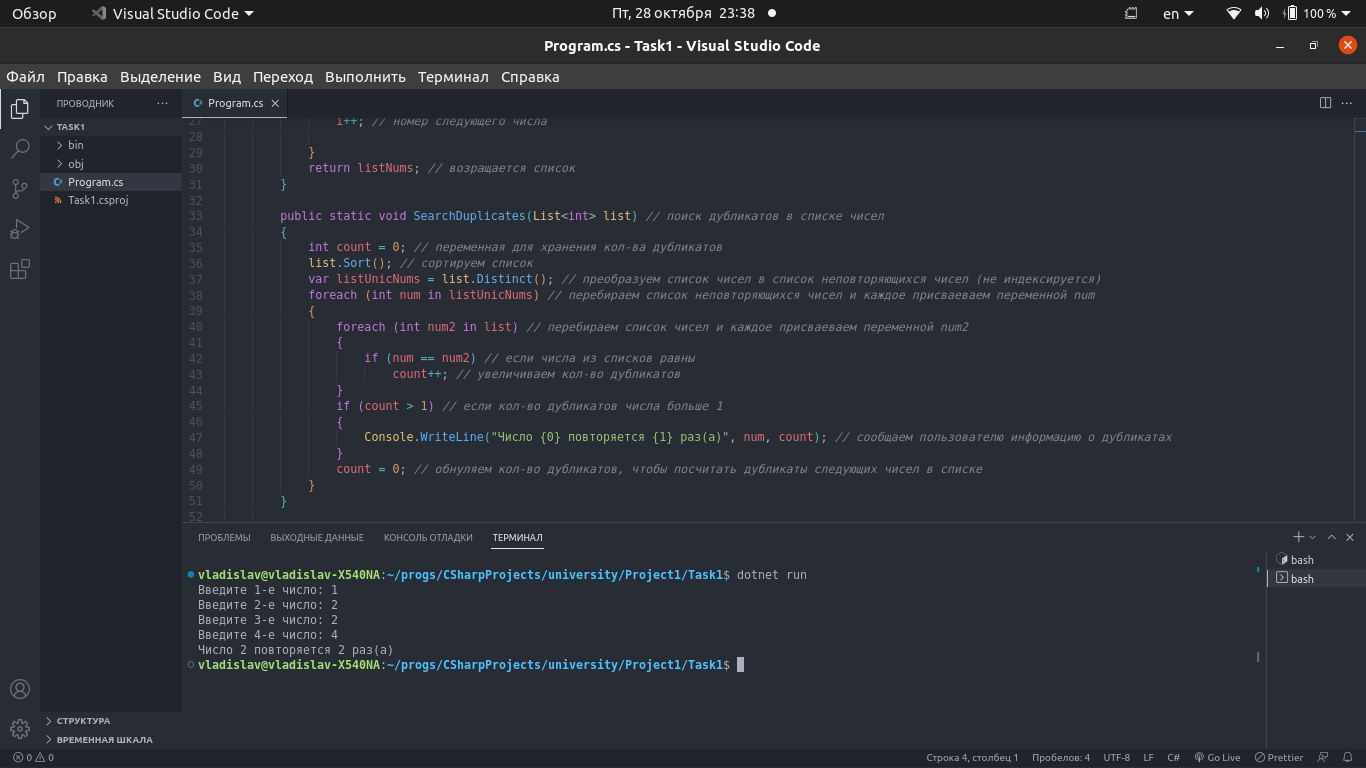
            SearchDuplicates(ListNums(4));

        }

    }

}

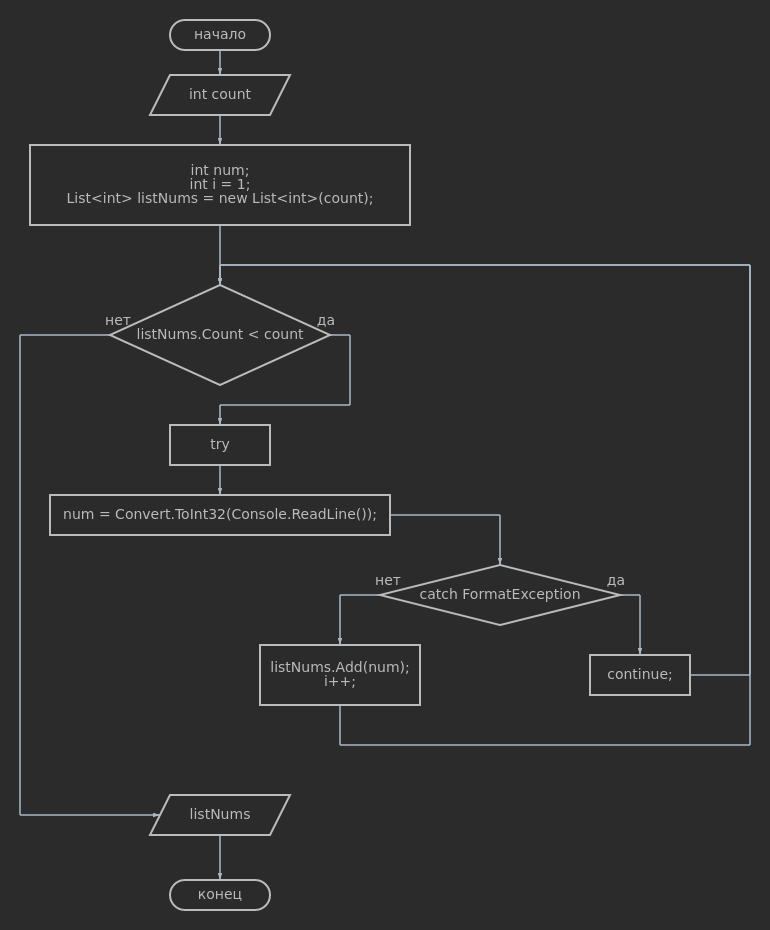
**Программа**

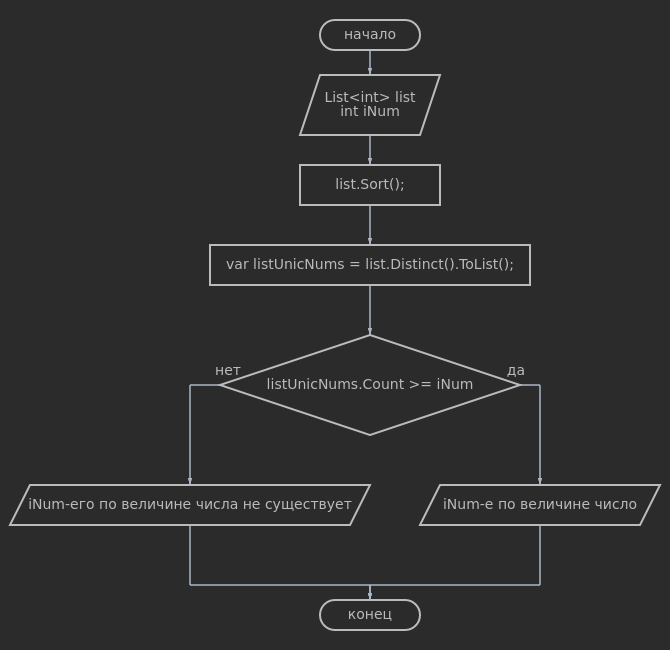
**Вывод результатов на экран**

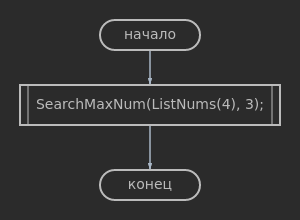
## Задание 2

Разработайте алгоритм, который запрашивает у пользователя четыре целых числа и затем находит третье по величине число, если оно существует. Нарисуйте блок-схему алгоритма. Программный код может быть реализован на одном из следующих языков программирования: С, С++, С#, Java.

**Блок-схема**

*diagram-ListNums*

*diagram-SearchMaxNum*

*diagram-Main*

### Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Task2

{

    class Program

    {

*public* *static* List<int> ListNums(int count) *// запрашивает у пользователя желаемое кол-во чисел и возвращет список введенных чисел*

        {

            int num; *// переменная в которой будут храниться по очередно вводимые числа*

            int i = 1; *// номер вводимого числа*

            List<int> listNums = new List<int>(count); *// список в котором будут храниться числа*

            while (listNums.Count < count) *// пока длина списка не равна желаемому кол-ву чисел*

            {

                try *// пытаемся привести введенное число к типу данных int*

                {

                    Console.Write("Введите {0}-е число: ", i);

                    num = Convert.ToInt32(Console.ReadLine()); *// присваеваем переменной введенное число и преобразуем в тип данных int*

                }

                catch (FormatException) *// если поймана ошибка при попытке преобразовать число*

                {

                    Console.WriteLine("Ожидается число...");

                    continue; *// цикл запускается заново*

                }

                listNums.Add(num); *// добавляем введенное число в список*

                i++; *// номер следующего числа*

            }

            return listNums; *// возращается список*

        }

*/\* поиск наибольшего числа*

*(принимает список чисел и индекс числа по величине, которое нужно найти)\*/*

*public* *static* void SearchMaxNum(List<int> list, int iNum)

        {

            list.Sort(); *// сортируем список*

*/\*преобразуем список чисел в список неповторяющихся чисел (не индексирутся),*

*преобразуем cонова в список\*/*

            var listUnicNums = list.Distinct().ToList();

            if (listUnicNums.Count >= iNum)

            {

                System.Console.WriteLine("{0}-е по величине число: {1}", iNum, listUnicNums[iNum - 1]);

            }else{

                System.Console.WriteLine("{0}-е по величине число: не существует", iNum);

            }

        }

*static* void Main(string[] args)

        {

*/\**

*Запрашиваем у пользователя четыре целых числа*

*и затем находим третье по величине число, если оно существует.*

*\*/*

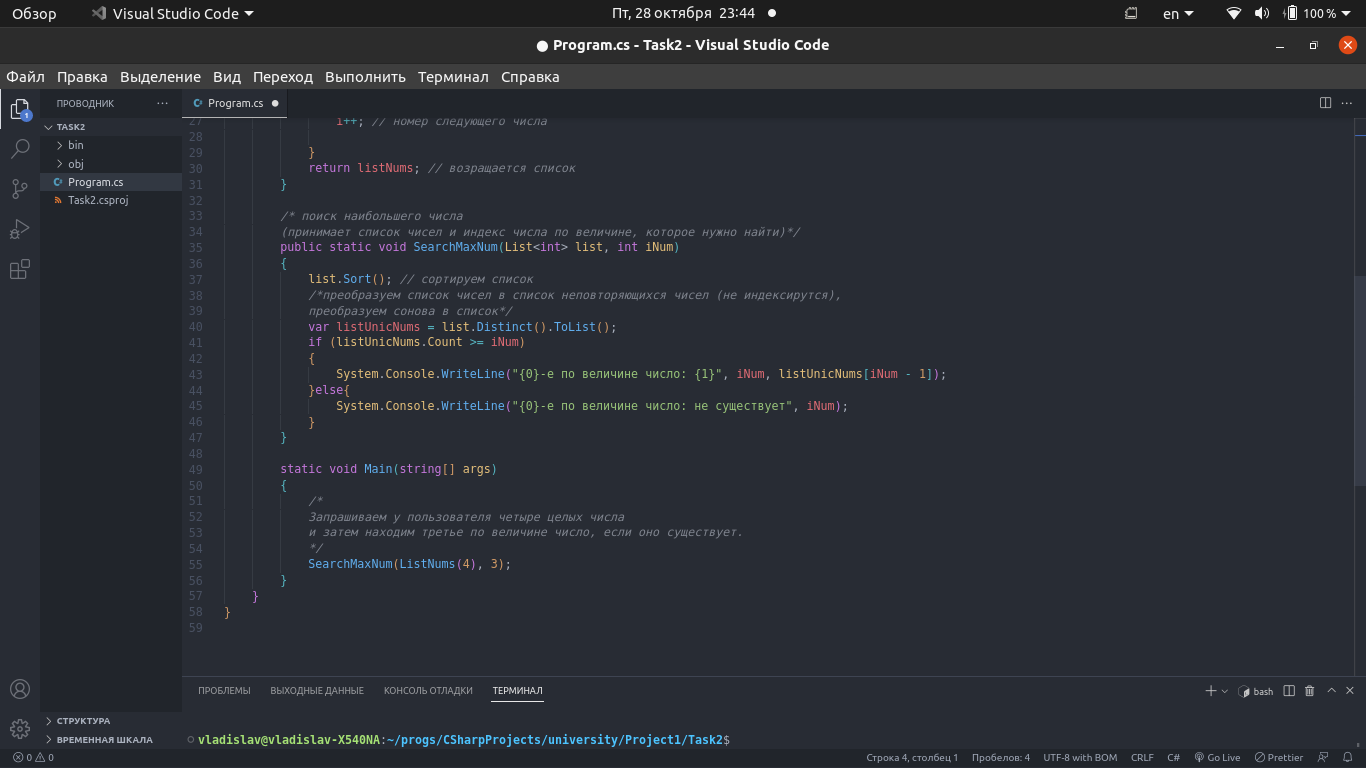
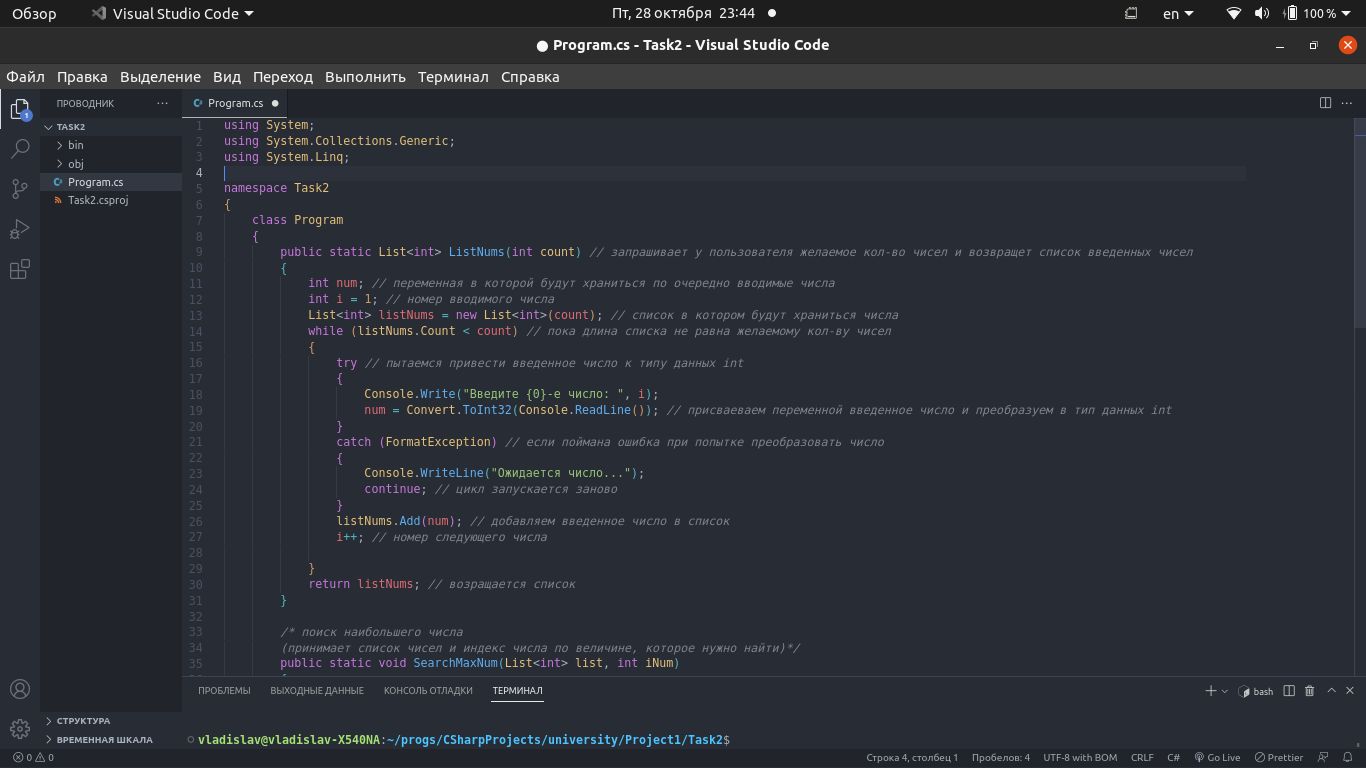
            SearchMaxNum(ListNums(4), 3);

        }

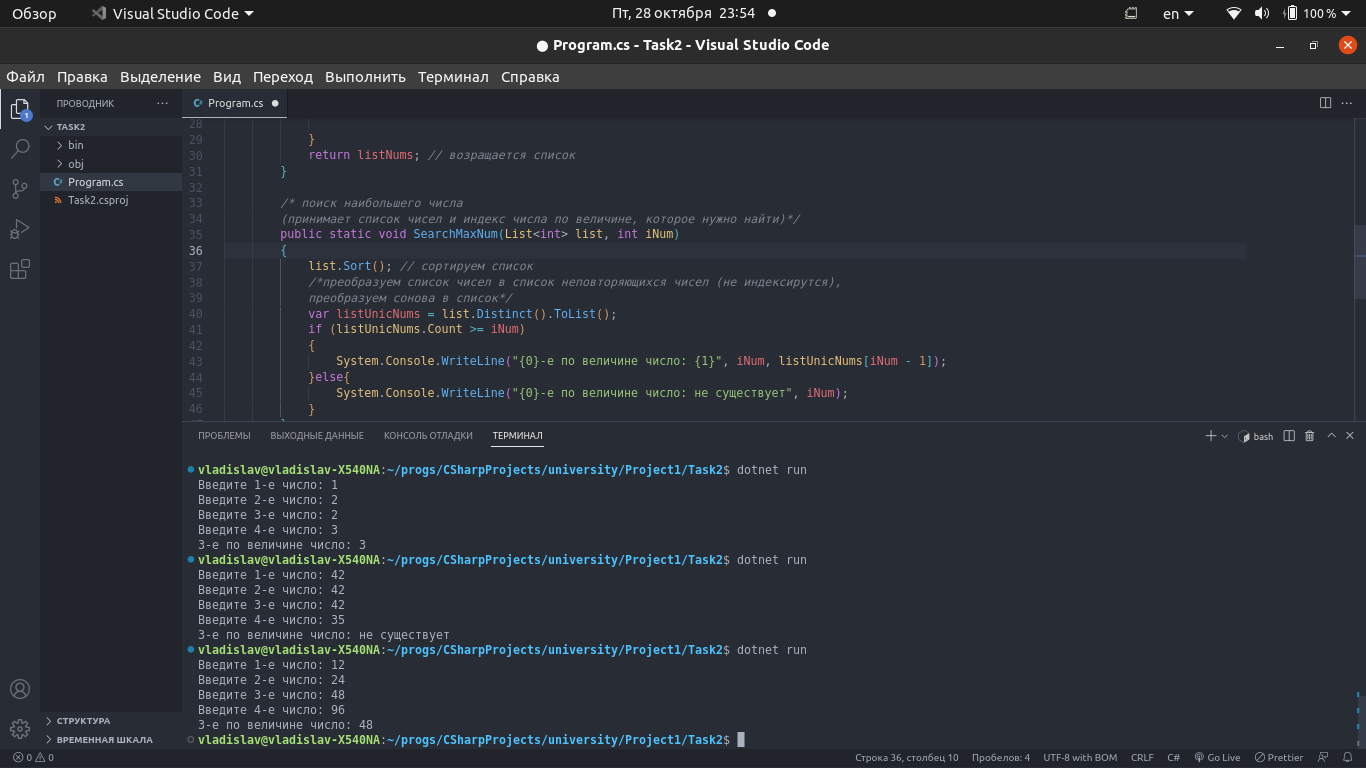
    }

}

**Программа**

****

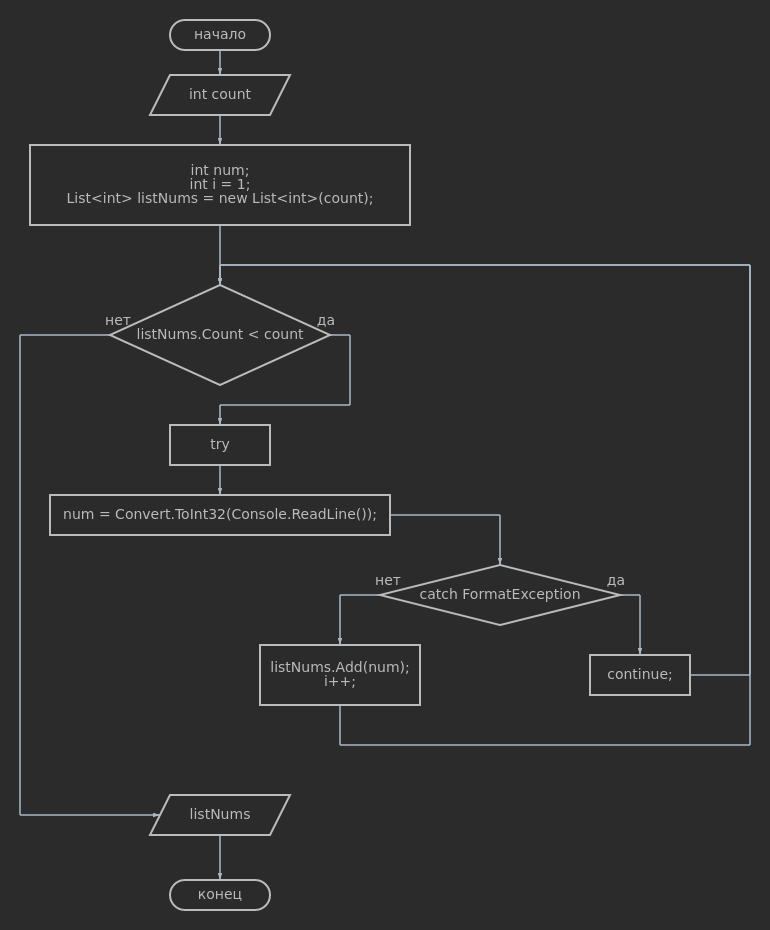
**Вывод результатов на экран**

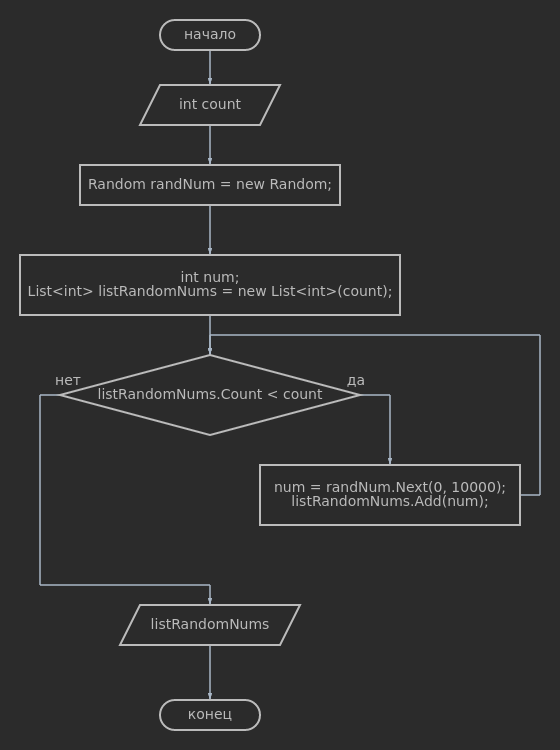


## Задание 3

Разработайте алгоритм, который случайным образом задает четыре целых числа и затем определяет максимальное и минимальное число, а также находит количество максимальных и минимальных чисел среди введенных. Нарисуйте блок-схему алгоритма. Программный код может быть реализован на одном из следующих языков программирования: С, С++, С#, Java.

**Блок-схема**

*diagram-ListNums*

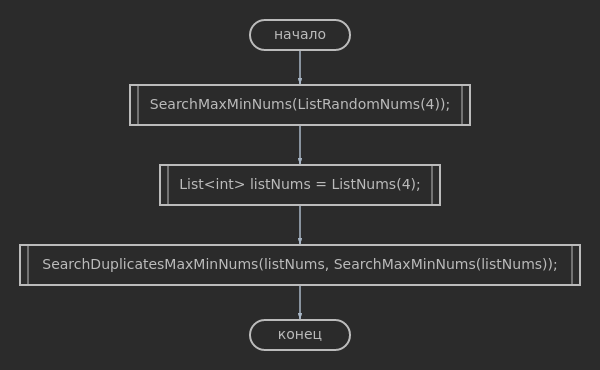
*diagram-ListRandomNums*

****

*diagram-SearchMaxMinNums*

****

*diagram-SearchDuplicatesMaxMinNums*

*diagram-Main*

### Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Task3

{

    class Program

    {

*public* *static* List<int> ListNums(int count) *// запрашивает у пользователя желаемое кол-во чисел и возвращет список введенных чисел*

        {

            int num; *// переменная в которой будут храниться по очередно вводимые числа*

            int i = 1; *// номер вводимого числа*

            List<int> listNums = new List<int>(count); *// список в котором будут храниться числа*

            while (listNums.Count < count) *// пока длина списка не равна желаемому кол-ву чисел*

            {

                try *// пытаемся привести введенное число к типу данных int*

                {

                    Console.Write("Введите {0}-е число: ", i);

                    num = Convert.ToInt32(Console.ReadLine()); *// присваеваем переменной введенное число и преобразуем в тип данных int*

                }

                catch (FormatException) *// если поймана ошибка при попытке преобразовать число*

                {

                    Console.WriteLine("Ожидается число...");

                    continue; *// цикл запускается заново*

                }

                listNums.Add(num); *// добавляем введенное число в список*

                i++; *// номер следующего числа*

            }

            return listNums; *// возращается список*

        }

*public* *static* List<int> ListRandomNums(int count) *// рандомный список чисел*

        {

            Random randNum = new Random(); *// cоздание объекта для генерации чисел*

            int num;

            List<int> listRandomNums = new List<int>(count); *// список в котором будут храниться числа*

            while (listRandomNums.Count < count) *// пока длина списка не равна желаемому кол-ву чисел*

            {

                num = randNum.Next(0, 10000);

                listRandomNums.Add(num); *// добавляем введенное число в список*

            }

            Console.WriteLine(string.Join(", ", listRandomNums));

            return listRandomNums; *// возращается список*

        }

*public* *static* void SearchDuplicatesMaxMinNums(List<int> list, List<int> list2)

        {

            int count = 0; *// переменная для хранения кол-ва дубликатов*

            list.Sort(); *// сортируем список*

            foreach (int num in list2) *// перебираем список неповторяющихся чисел и каждое присваеваем переменной num*

            {

                foreach (int num2 in list) *// перебираем список чисел и каждое присваеваем переменной num2*

                {

                    if (num == num2) *// если числа из списков равны*

                        count++; *// увеличиваем кол-во дубликатов*

                }

                Console.WriteLine("Число {0} повторяется {1} раз(а)", num, count); *// сообщаем пользователю информацию о дубликатах*

                count = 0; *// обнуляем кол-во дубликатов, чтобы посчитать дубликаты следующих чисел в списке*

            }

        }

*/\* поиск максимального и минимального числа\*/*

*public* *static* List<int> SearchMaxMinNums(List<int> list)

        {

            List<int> listMaxMinNums = new List<int>(2);

            list.Sort(); *// сортируем список*

            if (list[0] != list[list.Count - 1]){

                listMaxMinNums.Add(list[0]);

                listMaxMinNums.Add(list[list.Count - 1]);

                System.Console.WriteLine("Минимальное число: {0}", listMaxMinNums[0]);

                System.Console.WriteLine("Максимальное число: {0}", listMaxMinNums[1]);

            }else{

                listMaxMinNums.Add(list[0]);

                System.Console.WriteLine("Минимальное и максимальное число: {0}", listMaxMinNums[0]);

            }

            return listMaxMinNums;

        }

*static* void Main(string[] args)

        {

*/\**

*Cлучайным образом задается четыре целых числа*

*затем определяется максимальное и минимальное число,*

*также находится количество максимальных и минимальных чисел среди введенных.*

*\*/*

            SearchMaxMinNums(ListRandomNums(4));

            List<int> listNums = ListNums(4);

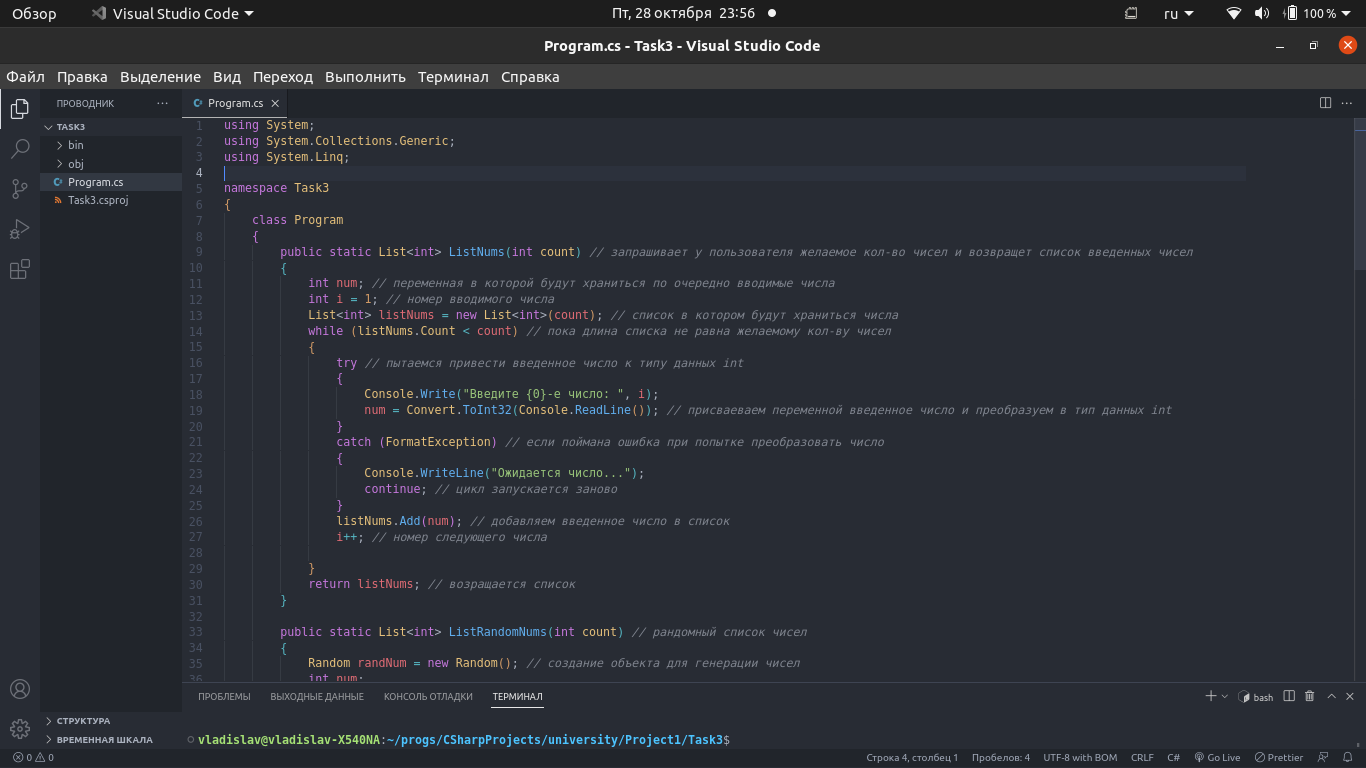
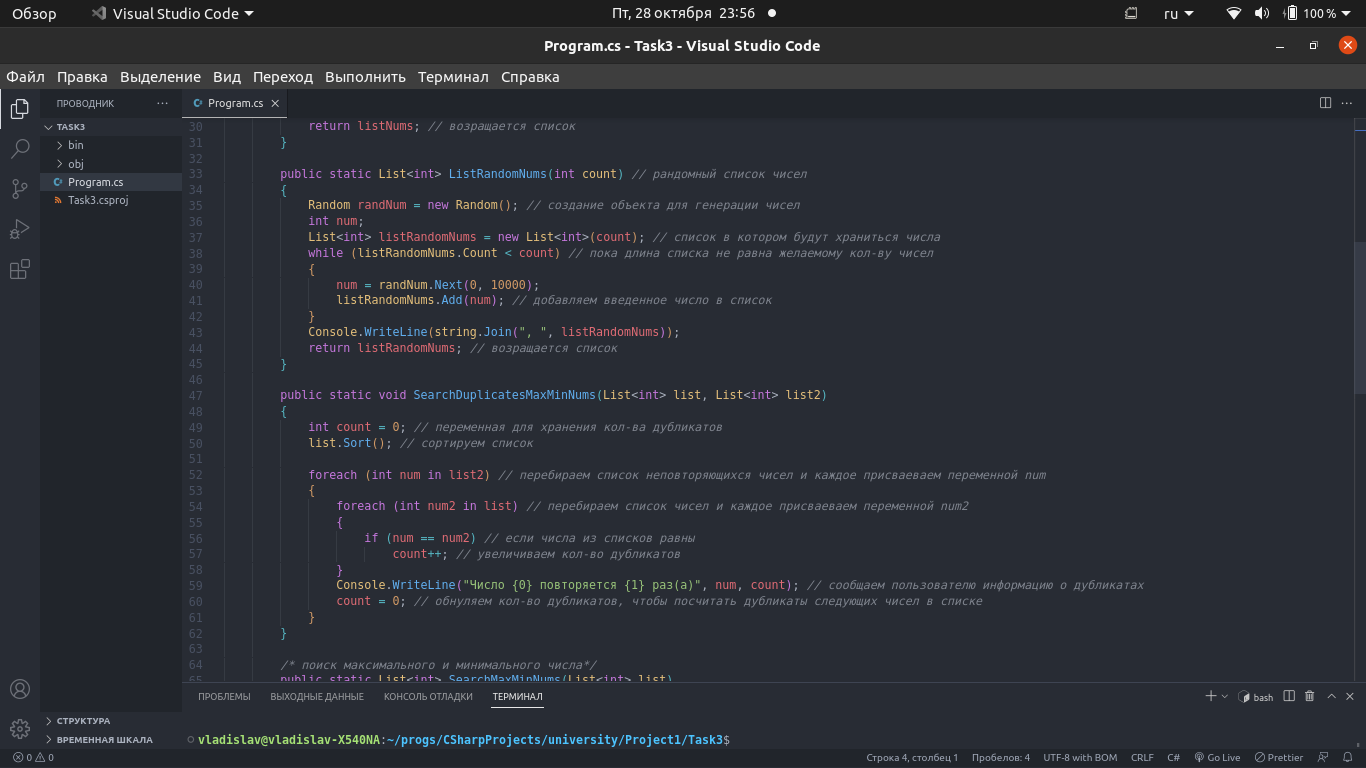
            SearchDuplicatesMaxMinNums(listNums, SearchMaxMinNums(listNums));

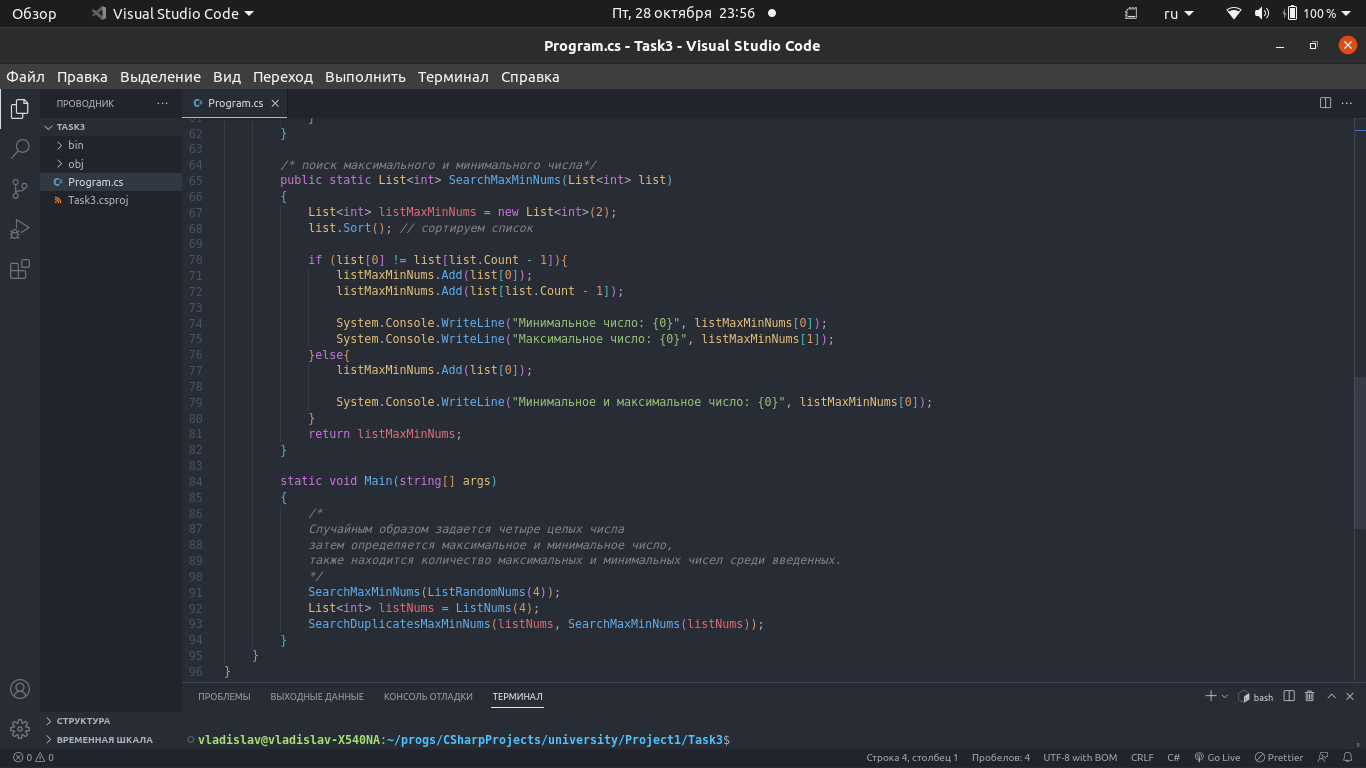
        }

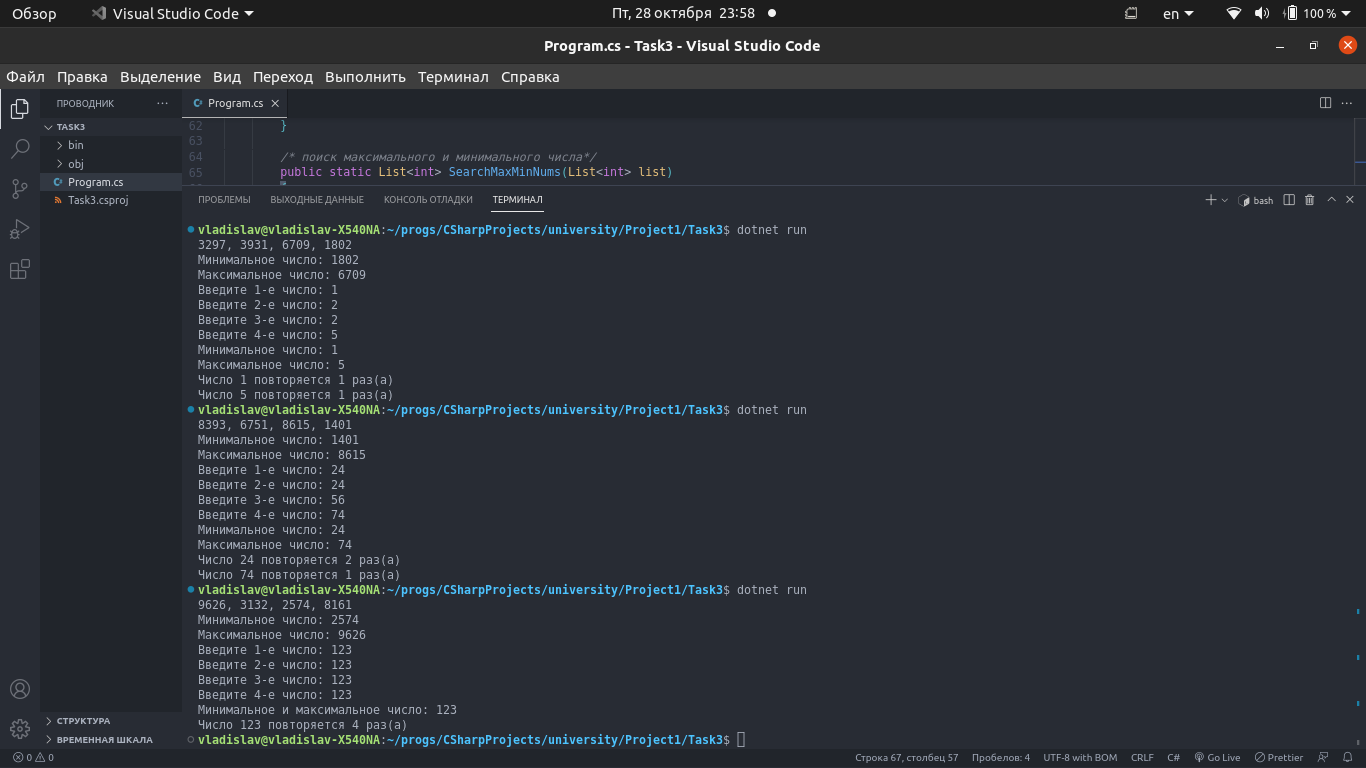
    }

}

**Программа**

****

****

**Вывод результатов на экран**

# Практическое задание 2

## Задание 1

Создайте одномерный массив. Выведите его содержимое на экран. Выполните реверс содержимого массива (переверните наоборот: первый элемент меняется местами с последним, второй с предпоследним и т.д.). Выведите содержимое на экран.

### Program.cs

class Program

{

*// перевернуть массив*

*public* *static* int[] FlipArray(int[] arr)

    {

        int[] arr1 = new int[arr.Length];

        int j = 0;

        for (int i = arr.Length - 1; i >= 0; i--)

        {

            arr1[j] = arr[i];

            j++;

        }

        return arr1;

    }

*private* *static* void Main(string[] args)

    {

        int[] arr = new int[5] { 1, 2, 3, 4, 5 };

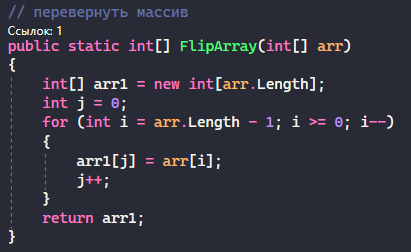
        Console.WriteLine(string.Join(", ", arr));

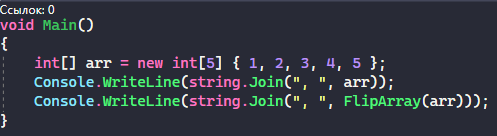
        Console.WriteLine(string.Join(", ", FlipArray(arr)));

    }

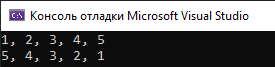
}

**Программа**

****

****

**Вывод результатов на экран**

****

## Задание 2

Записать -1 вместо минимального элемента массива X(11) и +1 - вместо максимального элемента.

### Program.cs

class Program

{

*// массив случайных чисел*

*static* int[] RandomArr(int length)

    {

        int[] arr = new int[length];

        Random randNum = new Random();

        int num;

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

        {

            num = randNum.Next(-1000, 1000);

            arr[i] = num;

        }

        return arr;

    }

*// наименьшее и наибольшее число в массиве*

*static* int[] MinMaxNumsInArr(int[] arr)

    {

        int minNum = arr[0];

        int maxNum = arr[0];

        int[] arrMinMaxNums = new int[2];

        foreach (int num in arr)

        {

            if (minNum > num)

            {

                minNum = num;

            }

            if (maxNum < num)

            {

                maxNum = num;

            }

        }

        arrMinMaxNums[0] = minNum;

        arrMinMaxNums[1] = maxNum;

        return arrMinMaxNums;

    }

*// замена чисел*

*static* int[] ChangeNums(int[] arr, int[] arrMinMaxNums)

    {

        int[] newArr = new int[arr.Length];

        for (int i = 0; i < arr.Length; i++)

        {

            if (arrMinMaxNums[0] == arr[i])

            {

                newArr[i] = -1;

            }

            else if (arrMinMaxNums[1] == arr[i])

            {

                newArr[i] = 1;

            }

            else

            {

                newArr[i] = arr[i];

            }

        }

        return newArr;

    }

*static* void Main(string[] args)

    {

        int[] arr = RandomArr(11);

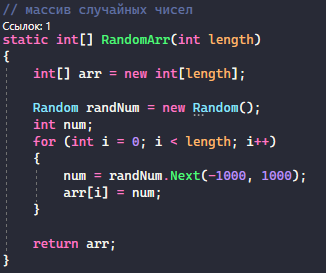
        Console.WriteLine("Array: {0}", string.Join(", ", arr));

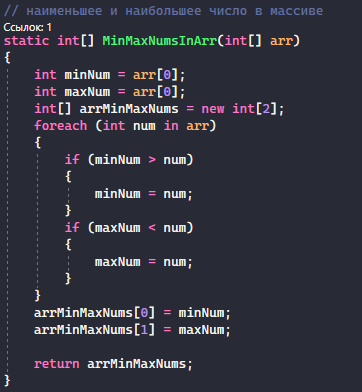
        Console.WriteLine("NewArray: {0}", string.Join(", ", ChangeNums(arr, MinMaxNumsInArr(arr))));

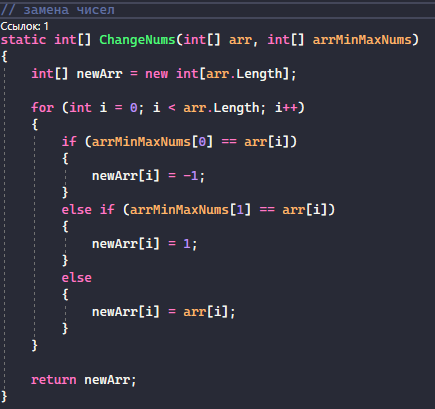
    }

}

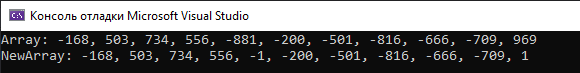
**Программа**

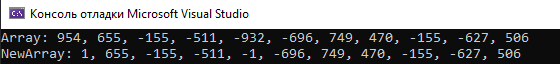
****

****

****

**Вывод результатов на экран**

****

****

## Задание 3

Дан массив A(32). Получить число отрицательных членов среди первых 20-ти элементов массива и количество нулевых элементов во всем массиве.

### Program.cs

namespace Task

{

*internal* class Program

    {

*static* int[] RandomArr(int length)

        {

            int[] randArr = new int[length];

            Random randNum = new Random();

            int num;

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

            {

                num = randNum.Next(-10, 10);

                randArr[i] = num;

            }

            return randArr;

        }

*// numOfEl - кол-ов элементов которые нужно обойти*

*static* int NumOfNegativeEl(int[] arr, int numOfEl)

        {

            int num = 0;

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

            {

                if (arr[i] < 0)

                {

                    num++;

                }

            }

            return num;

        }

*static* int NumOfZeroEl(int[] arr)

        {

            int num = 0;

            for (int i = 0; i < arr.Length; i++)

            {

                if (arr[i] == 0)

                {

                    num++;

                }

            }

            return num;

        }

*static* void Main(string[] args)

        {

            int[] arr = RandomArr(32);

            Console.WriteLine("Элементы массива: {0}", string.Join(", ", arr));

            Console.WriteLine(

                "Кол-во отприцательный чисел среди первых 20 элементов массива: {0}",

                NumOfNegativeEl(arr, 20)

                );

            Console.WriteLine(

                "Кол-во нулевых элементов массива: {0}",

                NumOfZeroEl(arr)

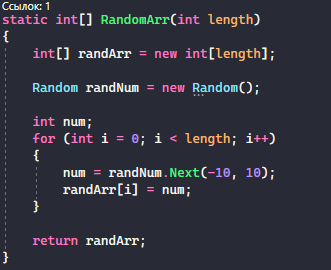
                );

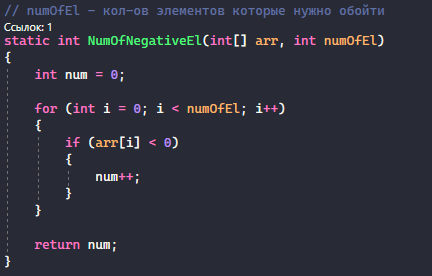
        }

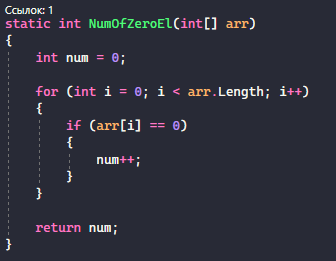
    }

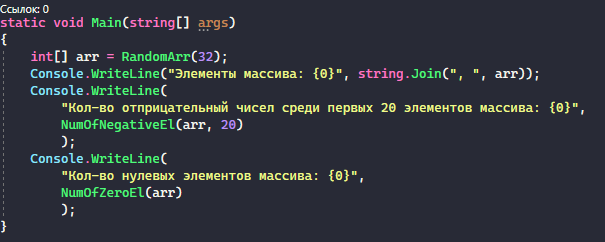
}

**Программа**

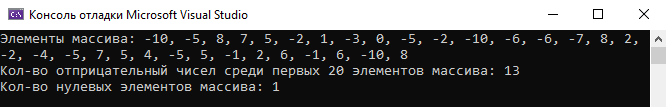
****

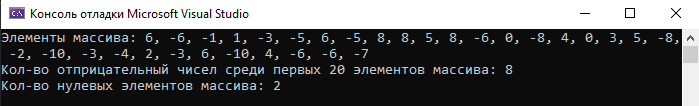
****

****

****

**Вывод результатов на экран**

****

****

## Задание 4

Четные строки матрицы K(8,3) заменить на значение Н.

### Program.cs

namespace Task

{

*internal* class Program

    {

*// row - строка*

*// col - столбец*

*static* object[,] SetMatrix(int rowsNum, int colsNum)

        {

            object[,] matrix = new object[rowsNum, colsNum];

            Random randNum = new Random();

            int num;

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

            {

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

                {

                    num = randNum.Next(-10, 10);

                    matrix[i, j] = num;

                }

            }

            return matrix;

        }

*// newEl - новый элемент на который будут заменены четные строки*

*static* object[,] ChangeElOfMatrix(object[,] matrix, char newEl)

        {

            int rowsNum = matrix.GetLength(0);

            int colsNum = matrix.GetLength(1);

            object[,] newMatrix = new object[rowsNum, colsNum];

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

            {

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

                {

                    if ((i + 1) % 2 == 0)

                    {

                        newMatrix[i, j] = newEl;

                    }

                    else

                    {

                        newMatrix[i, j] = matrix[i, j];

                    }

                }

            }

            return newMatrix;

        }

*static* void Show2DMatrix(object[,] matrix)

        {

            string rowSrting = "";

            var lengthRow = 0;

            string delim;

            int rowsNum = matrix.GetLength(0);

            int colsNum = matrix.GetLength(1);

            Console.WriteLine("Matrix Params:\n\trow = {0};\n\tcolumn = {1};", rowsNum, colsNum);

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

            {

                rowSrting = "";

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

                {

                    rowSrting += $"{matrix[i, j],4}{"",-4}| ";

                }

                lengthRow = rowSrting.Length;

                delim = string.Join("", Enumerable.Repeat("-", lengthRow));

                Console.Write(delim + "\n|" + rowSrting + "\n");

            }

            delim = string.Join("", Enumerable.Repeat("-", lengthRow));

            Console.WriteLine(delim);

        }

*static* void Main(string[] args)

        {

            object[,] matrix = SetMatrix(8, 3);

            Show2DMatrix(matrix);

            Console.WriteLine("\nЗаменяем четные строки  матрицы на 'H'");

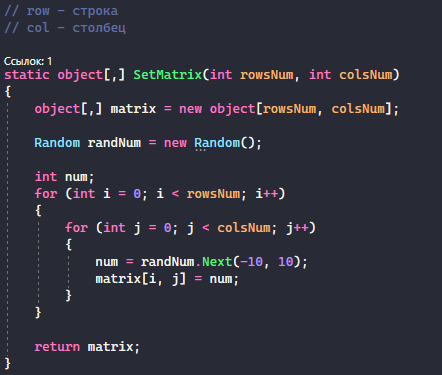
            Show2DMatrix(ChangeElOfMatrix(matrix, 'H'));

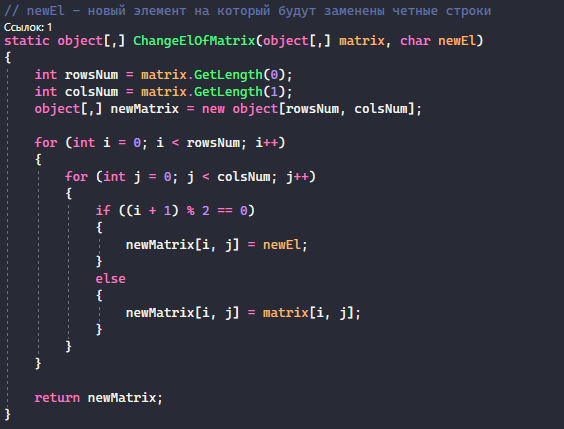
        }

    }

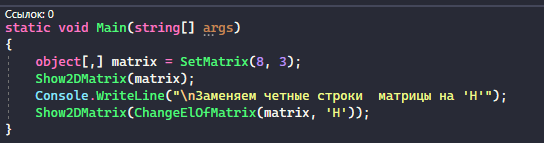
}

**Программа**

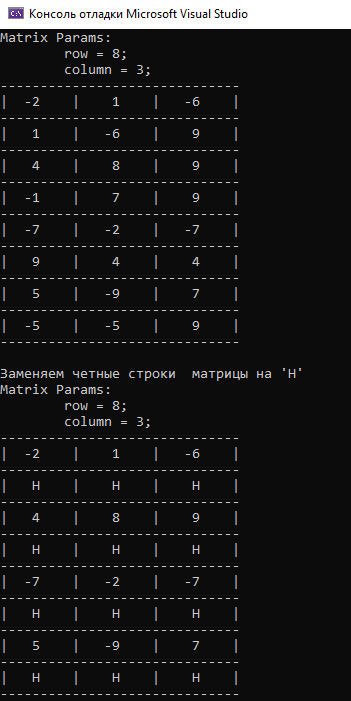
****

****

****

****

**Вывод результатов на экран**

****

## Задание 5

Среди элементов, расположенных выше побочной диагонали матрицы R(6,6), найти те элементы, которые удовлетворяют условию K2 <= Rij <= K1 (K1, K2 - произвольные числа), и сформировать из них одномерный массив.

### Program.cs

namespace Task

{

*internal* class Program

    {

*static* int[,] SetMatrix(int rowsNum, int colsNum)

        {

            int[,] matrix = new int[rowsNum, colsNum];

            Random randNum = new Random();

            int num;

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

            {

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

                {

                    num = randNum.Next(-10, 10);

                    matrix[i, j] = num;

                }

            }

            return matrix;

        }

*/\* получить элементы удов условию:*

*Среди элементов, расположенных выше побочной диагонали матрицы R(6,6)*

*найти те элементы,*

*которые удовлетворяют условию K2 <= Rij <= K1 (K1, K2 - произвольные числа),*

*и сформировать из них одномерный массив \*/*

*static* int[] GetEl(int[,] matrix, int k1, int k2)

        {

            int rowsNum = matrix.GetLength(0);

            int colsNum = matrix.GetLength(1);

            int n = rowsNum - 1;

            int length = 0;

*// кол-во элементов удов условию*

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

            {

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

                {

                    if (i + j < n)

                    {

                        if (matrix[i, j] <= k1 && k2 <= matrix[i, j])

                        {

                            length++;

                        }

                    }

                }

            }

            int[] arr = new int[length];

            int l = 0;

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

            {

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

                {

                    if (i + j < n)

                    {

                        if (matrix[i, j] <= k1 && k2 <= matrix[i, j])

                        {

                            arr[l] = matrix[i, j];

                            l++;

                        }

                    }

                }

            }

            return arr;

        }

*// вывод матрицы на экран*

*static* void Show2DMatrix(int[,] matrix)

        {

            string rowString = "";

            var lengthRow = 0;

            string delim;

            int rowsNum = matrix.GetLength(0);

            int colsNum = matrix.GetLength(1);

            Console.WriteLine("Matrix Params:\n\trow = {0};\n\tcolumn = {1};", rowsNum, colsNum);

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

            {

                rowString = "";

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

                {

                    rowString += $"{matrix[i, j],4}{"",-4}| ";

                }

                lengthRow = rowString.Length;

                delim = string.Join("", Enumerable.Repeat("-", lengthRow));

                Console.Write(delim + "\n|" + rowString + "\n");

            }

            delim = string.Join("", Enumerable.Repeat("-", lengthRow));

            Console.WriteLine(delim);

        }

*static* void Main(string[] args)

        {

            int[,] matrix = SetMatrix(8, 8);

            Show2DMatrix(matrix);

            Console.WriteLine(

                "Элементы, расположенные выше побочной диагонали матрицы, которые удовлетворяют условию -5 <= Rij <= 5: {0}",

                string.Join(", ", GetEl(matrix, 5, -5))

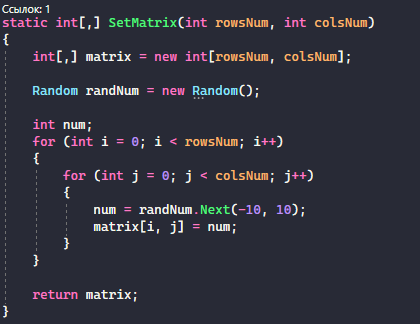
            );

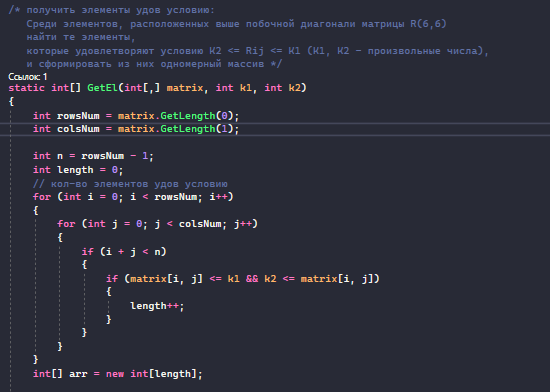
        }

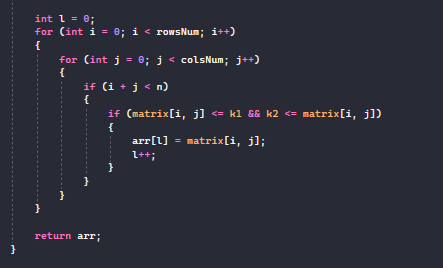
    }

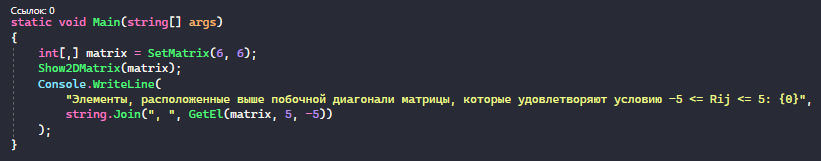
}

**Программа**

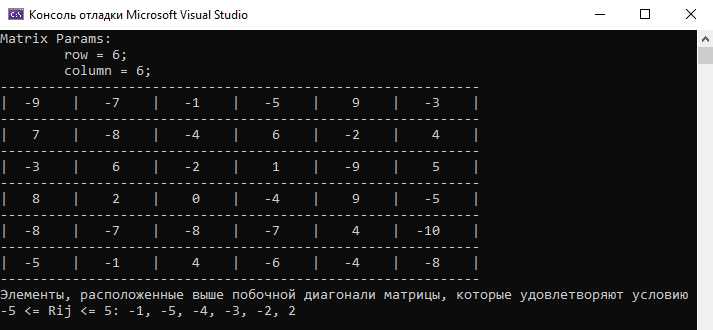
****

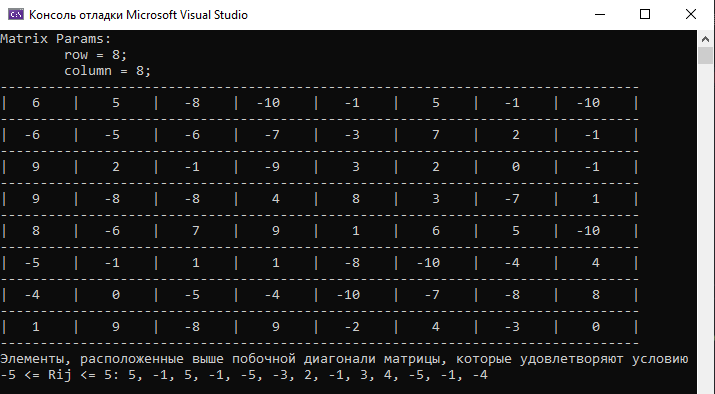
****

****

****

**Вывод результатов на экран**

****

****

## Задание 6

Создайте двумерный массив А(N,M). Значения N и M вводит пользователь с клавиатуры. Элементы массива могут принимать 2 значения – 0 и 1. Количество 1 (K) определяет пользователь. Эти 1 распределяются по массиву случайнам образом. Должно получиться на подобие следующего:

N=3, M=5, K=4

00100

10010

00100

На экране вывести картинку массива и его зеркальное отображение по всем направлениям.

Массив будет только один исходный. Для вывода на экран используйте вложенный цикл. За счет правильного выбора индексов (счетчиков массива) выводятся зеркальные копии исходного массива. При выводе на экран используйте пробел, если значение массива равно 0, и \*, если значение – 1.

### Program.cs

namespace Task

{

*internal* class Program

    {

*// задать матрицу*

*static* int[,] SetMatrix2D(int rowsNum, int colsNum, int k)

        {

            int[,] matrix = new int[rowsNum, colsNum];

            Random randNum = new Random();

            int count = k; *// кол-во единиц*

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

            {

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

                {

                    if (count > 0)

                    {

                        matrix[i, j] = randNum.Next(0, 2);

                        if (matrix[i, j] == 1)

                            count--;

                    }

                    else

                    {

                        matrix[i, j] = 0;

                    }

                }

            }

            if (count != 0)

            {

                Console.WriteLine("Перезапись матрицы");

                matrix = SetMatrix2D(rowsNum, colsNum, k);

            }

            return matrix;

        }

*// вывод матрицы в консоль*

*static* void Show2DMatrix(int[,] matrix)

        {

            int rowsNum = matrix.GetLength(0); *// кол-во столбцов в матрице*

            int colsNum = matrix.GetLength(1); *// кол-во строк в матрице*

*// Console.WriteLine("{0}, {1}", col, row);*

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

            {

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

                {

                    Console.Write($"{matrix[i, j]} ");

                }

                Console.WriteLine();

            }

        }

*// вывод картинки*

*static* void Show2DMatrixPic(int[,] matrix)

        {

            int rowsNum = matrix.GetLength(0); *// кол-во столбцов в матрице*

            int colsNum = matrix.GetLength(1); *// кол-во строк в матрице*

*// Console.WriteLine("{0}, {1}", col, row);*

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

            {

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

                {

                    if (matrix[i, j] == 0)

                        Console.Write(" ");

                    else

                        Console.Write("\*");

                }

                Console.WriteLine();

            }

        }

*// вывод матрицы по всем направлениям*

*static* int[,] MatrixReflection(int[,] matrix)

        {

            int rowsNum = matrix.GetLength(0); *// кол-во строк в матрице*

            int colsNum = matrix.GetLength(1); *// кол-во столбцов в матрице*

            int[,] newMatrix = new int[rowsNum \* 2, colsNum \* 2]; *// новая матрица*

*// Console.WriteLine("{0}, {1}", col, row);*

            Console.WriteLine();

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

            {

*// без отражения*

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

                {

                    newMatrix[i, j] = matrix[i, j];

                    Console.Write($"{newMatrix[i, j]} ");

                }

                Console.Write("|"); *// разделитель по вертикали*

*// отражение строк*

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

                {

                    newMatrix[i, colsNum + j] = matrix[i, colsNum - 1 - j];

                    Console.Write($"{newMatrix[i, colsNum + j]} ");

                }

                Console.WriteLine();

            }

*// разделитель по горизонтали*

            for (int i = 0; i <= rowsNum \* 2; i++)

            {

                Console.Write("--");

            }

            Console.WriteLine();

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

            {

*// отражение столбцов*

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

                {

                    newMatrix[rowsNum + i, j] = matrix[rowsNum - 1 - i, j];

                    Console.Write($"{newMatrix[rowsNum + i, j]} ");

                }

                Console.Write("|"); *// разделитель по вертикали*

*// отражение столбцов и строк*

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

                {

                    newMatrix[rowsNum + i, colsNum + j] = matrix[rowsNum - 1 - i, colsNum - 1 - j];

                    Console.Write($"{newMatrix[rowsNum + i, colsNum + j]} ");

                }

                Console.WriteLine();

            }

            Console.WriteLine();

            return newMatrix;

        }

*static* void Main(string[] args)

        {

            int n = 0;

            while (n <= 0)

            {

                Console.Write("Введите кол-во строк (больше 0): ");

                try

                {

                    n = Convert.ToInt32(Console.ReadLine());

                }

                catch (FormatException)

                {

                    Console.WriteLine("Ожидается число");

                    continue;

                }

            }

            int m = 0;

            while (m <= 0)

            {

                Console.Write("Введите кол-во столбцов (больше 0): ");

                try

                {

                    m = Convert.ToInt32(Console.ReadLine());

                }

                catch (FormatException)

                {

                    Console.WriteLine("Ожидается число");

                    continue;

                }

            }

            int k = 0;

            int maxNum = (n \* m) / 2;

            while (k <= 0 || k > maxNum)

            {

                Console.Write($"Введите кол-во единиц (больше 0 и не больше половины размера матрицы - {maxNum}): ");

                try

                {

                    k = Convert.ToInt32(Console.ReadLine());

                }

                catch (FormatException)

                {

                    Console.WriteLine("Ожидается число");

                    continue;

                }

            }

            Console.WriteLine($"n = {n}; m = {m}; k = {k}");

            int[,] matrix = SetMatrix2D(n, m, k);

            Show2DMatrix(matrix);

            int[,] newMatrix = MatrixReflection(matrix);

            Show2DMatrixPic(newMatrix);

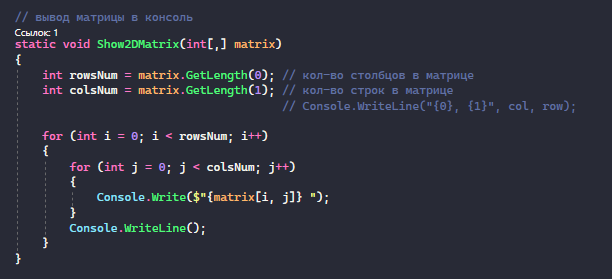
        }

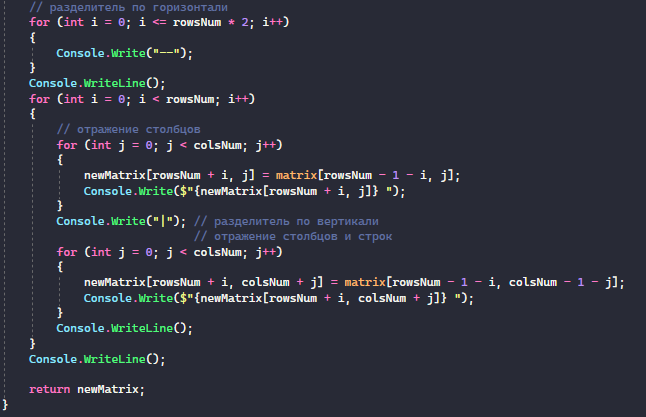
    }

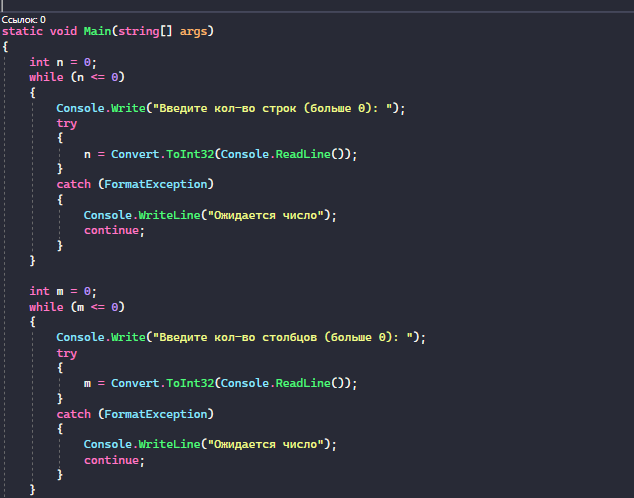
}

**Программа**

****

****

****

****

****

**Вывод результатов на экран**

****

## Задание 7

Создайте одномерный массив, который содержит полный набор возможных значений типа UInt16, кроме одного. Определите, какого числа не хватает.

### Program.cs

namespace Task

{

*internal* class Program

    {

*// создание массива*

*static* ushort[] CreateArr(ushort length)

        {

*// массив uint16 чисел с неполным набором значений*

            ushort[] arr = new ushort[length];

*// рандомное число не входящее в массив*

            Random randNum = new Random();

            ushort numException = Convert.ToUInt16(randNum.Next(0, length + 1));

*// число добавляемое в массив*

            ushort num = 0;

*// наполнение массива числами*

            for (ushort i = 0; i < length; i++)

            {

                if (num == numException)

                    num++;

                arr[i] = num;

                num++;

            }

            return arr;

        }

*// поиск недостающего числа*

*static* void SearchNumException(ushort[] arr, ushort maxValue)

        {

            if (arr.Last() != maxValue)

            {

                Console.WriteLine($"Число найдено: {maxValue}");

            }

            else

            {

                for (ushort i = 0; i < maxValue; i++)

                {

                    if (arr[i] != i)

                    {

                        Console.WriteLine($"Число найдено: {i}");

                        break;

                    }

                }

            }

        }

*static* void Main(string[] args)

        {

*// ushort - uint16, хранит от 0 до 65535*

*// максимальное значение uint16*

            ushort maxValue = ushort.MaxValue;

            ushort[] arr = CreateArr(maxValue);

            Console.WriteLine("Массив наполнен\nОдного числа не хватает\nПоиск недостающего числа");

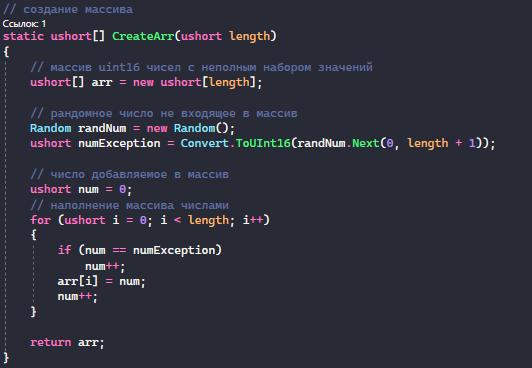
            SearchNumException(arr, maxValue);

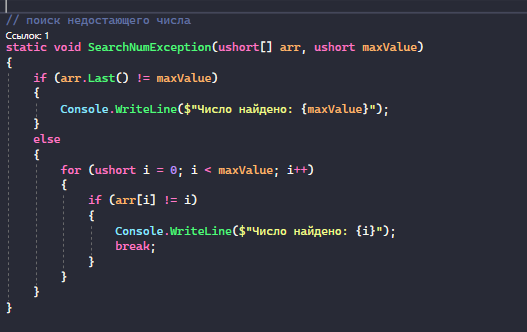
        }

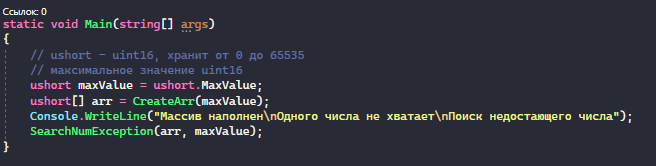
    }

}

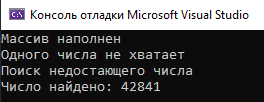
**Программа**

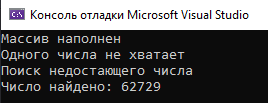
****

****

****

**Вывод результатов на экран**

****

****

# Практическое задание 3

## Задание 1

Написать единую программу на языке высокого уровня (С++, C#, Java), реализующую сортировку массива, заполненного случайными числами в диапазоне от 0 до 100, различными методами. Провести сравнительный анализ временной эффективности реализованных методов.

Должны быть реализованы следующие методы сортировки:

· Сортировка методом прямого обмена (сортировка методом пузырька).

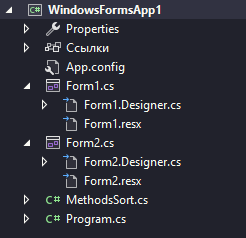
· Сортировка методом прямого включения.

· Сортировка методом прямого выбора.

· Шейкерная сортировка.

· Сортировка методом Шелла.

· Сортировка методом Хоара.

**Файлы программы**

**Код в файлах Program, Form1.Designer и Form2.Designer формируется автоматически**

### Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsApp1

{

internal static class Program

{

// компонентная объектная модель - component object model - COM

/\*

\* COM - это стандарт программирования, разработанный компанией Microsoft

\* Описывает правила создания программных компонентов,

\* при использовании которых нет необходимости учитывать,

\* в какой среде программирования создавался тот или иной модуль

\*/

/\*

\* Атрибут STAThreadAttribute используется для контроля потока

\* Атрибут обеспечивает механизм связи между

\* текущим потоком и другими потоками,

\* которые могут захотеть связаться с ним через COM.

\*/

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new MainForm());

}

}

}

### Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Runtime.CompilerServices;

using System.Security.Cryptography;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace WindowsFormsApp1

{

/// <summary>

/// главное окно

/// </summary>

public partial class MainForm : Form

{

#region главные переменные

int[] mainArr;

/\*

\* ................................

\* длина массива

\* начало диапазона

\* конец диапазона

\* ................................

\*/

// параметры массива

int arrLength;

int rangeFrom;

int rangeTo;

#endregion

#region форма

/// <summary>

/// инициализация компонентов формы

/// </summary>

public MainForm()

{

InitializeComponent();

// установка параметров графика среднего времени

setParamsChartMidTime(

"Анализ временной эффективности методов",

"Кол-во элементов",

"Среднее время сортировки массива (наносек.)"

);

}

#endregion

#region график

/// <summary>

/// установка параметров графика среднего времени

/// </summary>

/// <param name="mainTitle">заголовок графика</param>

/// <param name="xTitle">заголовок по оси X</param>

/// <param name="yTitle">заголовок по оси Y</param>

private void setParamsChartMidTime(string mainTitle, string xTitle, string yTitle)

{

// заголовок графика

chartMidTime.Titles.Add(mainTitle);

// заголовки осей

chartMidTime.ChartAreas[0].AxisX.Title = xTitle;

chartMidTime.ChartAreas[0].AxisY.Title = yTitle;

}

/// <summary>

/// добавить линию на график

/// </summary>

/// <param name="points">точки для отрисовик графика</param>

/// <param name="nameLine">название линии</param>

private void addLine(int[] points, string nameLine)

{

// создание линии

Series line = new Series(nameLine);

line.Points.DataBindY(points);

line.ChartType = SeriesChartType.Line;

line.BorderWidth = 2;

// добавление линии

try

{

chartMidTime.Series.Add(line);

}

catch (Exception)

{

addLine(points, nameLine + "\*");

}

}

/// <summary>

/// получить точки для отрисовки линии

/// </summary>

/// <param name="midTime">среднее время метода сортировки</param>

/// <returns>точки для отрисовки линии</returns>

private int[] getPoints(int midTime)

{

// задать точки

int[] points = new int[arrLength];

int y = 0;

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

{

points[i] = y;

y += midTime / arrLength;

}

return points;

}

#endregion

#region замер времени

/// <summary>

/// подсчет среднего времени

/// </summary>

/// <param name="arrTime">массив содержащий время сортировки на каждой операции</param>

/// <param name="numIters">число операций</param>

/// <returns>время сортировки (в наносек.)</returns>

private int calcMidTime(int[] arrTime, int numIters)

{

// сложение времени

int midTime = 0;

foreach (int time in arrTime)

midTime += time;

return midTime / numIters;

}

/// <summary>

/// получить время работы метода сортировки

/// </summary>

/// <param name="methodSort">метод сортировки</param>

/// <param name="arr">сортируемый массив</param>

/// <returns>время сортировки (в наносек.)</returns>

private int getTimeMethodSortNS(Action<int[]> methodSort)

{

// замер времени

Stopwatch stopWatch = new Stopwatch();

if (methodSort == null)

{

stopWatch.Start();

MethodsSort.quickSort(mainArr, 0, mainArr.Length - 1);

stopWatch.Stop();

}

else

{

stopWatch.Start();

methodSort(mainArr);

stopWatch.Stop();

}

return Convert.ToInt32(stopWatch.Elapsed.TotalMilliseconds \* 1000000);

}

/// <summary>

/// установить среднее время

/// </summary>

/// <param name="methodSort">метод сортировки</param>

/// <param name="numIters">число операций</param>

private void setMidTime(Action<int[]> methodSort, int numIters)

{

// проверка заданных данных

if (!checkData())

return;

// замер времени на каждой операции

int[] arrTime = new int[numIters];

List<string> arrStory = new List<string>();

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

{

mainArr = MethodsSort.getRandomArr(arrLength, rangeFrom, rangeTo);

System.Threading.Thread.Sleep(250);

if (methodSort == null)

arrTime[i] = getTimeMethodSortNS(null);

else

arrTime[i] = getTimeMethodSortNS(methodSort);

arrStory.Add($"[{string.Join(", ", mainArr)}]");

}

int midTime = calcMidTime(arrTime, numIters);

// сообщение о проведенной операции

MessageBox.Show($"Название: {getNameMethod()}\n" +

$"Длина массива: {arrLength}\n" +

$"Среднее время сортировки: {midTime}");

MessageBox.Show(string.Join("\n\n", arrStory));

// добавление линии на график

addLine(getPoints(midTime), $"{getNameMethod()}: {arrLength}");

}

#endregion

#region вводимые данные

/// <summary>

/// проверка заданных данных

/// </summary>

/// <returns>

/// true - если данные заданы верно

/// false - если данные заданы неверно

/// </returns>

private bool checkData()

{

try

{

arrLength = Convert.ToInt32(txtBoxLenArr.Text);

rangeFrom = Convert.ToInt32(txtBoxRangeFrom.Text);

rangeTo = Convert.ToInt32(txtBoxRangeTo.Text);

}

catch (FormatException)

{

MessageBox.Show("Неверно заданы значения");

return false;

}

// проверка диапазона

if (rangeFrom >= rangeTo)

{

MessageBox.Show("начало диапазона больше или равно концу диапазона");

return false;

}

return true;

}

/// <summary>

/// запись в текстовое поле (длина массива) только чисел

/// </summary>

private void txtBoxLenArr\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

/// <summary>

/// запись в текстовое поле (начало диапазона) только чисел

/// </summary>

private void txtBoxRangeFrom\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

/// <summary>

/// запись в текстовое поле (конец массива) только чисел

/// </summary>

private void txtBoxRangeTo\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = !char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar);

}

#endregion

#region поле выбора

/// <summary>

/// получить имя выбранного метода

/// </summary>

/// <returns>имя выбранного метода</returns>

private string getNameMethod()

{

return comboBoxSelectionMethodsSort.SelectedItem.ToString();

}

/// <summary>

/// получить индекс выбранного метода

/// </summary>

/// <returns>имя выбранного метода</returns>

private int getIndexMethod()

{

return comboBoxSelectionMethodsSort.SelectedIndex;

}

#endregion

#region кнопки

/// <summary>

/// открытие окна визуализации методов сортировки

/// </summary>

private void btnOpenVisualMethodsSortForm\_Click(object sender, EventArgs e)

{

int indexMethod = getIndexMethod();

if (indexMethod != -1)

{

VisualMethodsSortForm form = new VisualMethodsSortForm(

getNameMethod(), indexMethod);

form.Show();

}

else

{

MessageBox.Show("Метод не выбран");

}

}

/// <summary>

/// запустить вычисление среднего времени сортировки

/// </summary>

/// объект вызывающий событие, событие

private void btnMidTime\_Click(object sender, EventArgs e)

{

int indexMethod = getIndexMethod();

if (indexMethod == -1)

{

MessageBox.Show("Метод не выбран");

return;

}

MessageBox.Show($"Анализ врменной эффективности\n{getNameMethod()}");

switch (indexMethod)

{

case 0:

setMidTime(MethodsSort.bubbleSort, 10);

break;

case 1:

setMidTime(MethodsSort.insertionSort, 10);

break;

case 2:

setMidTime(MethodsSort.selectionSort, 10);

break;

case 3:

setMidTime(MethodsSort.shakerSort, 10);

break;

case 4:

setMidTime(MethodsSort.shellSort, 10);

break;

case 5:

setMidTime(null, 10);

break;

}

}

/// <summary>

/// запустить очистку графика

/// </summary>

/// объект вызывающий событие, событие

private void btnClearChartMidTime\_Click(object sender, EventArgs e)

{

chartMidTime.Series.Clear();

}

#endregion

}

}

### MethodsSort.cs

using System;

namespace WindowsFormsApp1

{

/// <summary>

/// работа с методами сортировки

/// </summary>

class MethodsSort {

/// <summary>

/// получить массив случаных чисел из диапазона

/// </summary>

/// <param name="len">желаемая длина массива</param>

/// <param name="from">начало диапазона</param>

/// <param name="to">конец диапазона (это число не входит в массив)</param>

/// <returns>массив случаных чисел из диапазона</returns>

public static int[] getRandomArr(int len, int from, int to)

{

// задается список уникальных случайных чисел из диапазона

int[] arr = new int[len];

Random randNum = new Random();

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

arr[i] = randNum.Next(from, to);

return arr;

}

/// <summary>

/// сортировка методом пузырька

/// </summary>

/// <param name="arr">сортируемый массив</param>

public static void bubbleSort(int[] arr)

{

for (int i = 0; i < arr.Length; i++)

{

for (int j = i + 1; j < arr.Length; j++)

{

// сравнение элементов

if (arr[i] > arr[j])

{

// обмен элементов

int buf = arr[i];

arr[i] = arr[j];

arr[j] = buf;

}

}

}

}

/// <summary>

/// сортировка методом прямого включения

/// </summary>

/// <param name="arr">сортируемый массив</param>

public static void insertionSort(int[] arr)

{

// buf // временная переменная для хранения значения элемента сортируемого массива

// item // индекс предыдущего элемента

for (int i = 1; i < arr.Length; i++)

{

int buf = arr[i]; // инициализируем временную переменную текущим значением элемента массива

int item = i - 1; // запоминаем индекс предыдущего элемента массива

while (item >= 0 && arr[item] > buf) // пока индекс не равен 0 и предыдущий элемент массива больше текущего

{

arr[item + 1] = arr[item]; // перестановка элементов массива

arr[item] = buf;

item--;

}

}

}

/// <summary>

/// сортировка методом прямого выбора

/// </summary>

/// <param name="arr">сортируемый массив</param>

public static void selectionSort(int[] arr)

{

for (int i = 0; i < arr.Length - 1; i++)

{

// поиск минимального числа

int min = i;

for (int j = i + 1; j < arr.Length; j++)

{

if (arr[j] < arr[min])

min = j;

}

// обмен элементов

int buf = arr[min];

arr[min] = arr[i];

arr[i] = buf;

}

}

/// <summary>

/// сортировка методом шейкера

/// </summary>

/// <param name="arr">сортируемый массив</param>

public static void shakerSort(int[] arr)

{

for (var i = 0; i < arr.Length / 2; i++)

{

var swapFlag = false;

//проход слева направо

for (var j = i; j < arr.Length - i - 1; j++)

{

if (arr[j] > arr[j + 1])

{

var temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

swapFlag = true;

}

}

//проход справа налево

for (var j = arr.Length - 2 - i; j > i; j--)

{

if (arr[j - 1] > arr[j])

{

var temp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = temp;

swapFlag = true;

}

}

//если обменов не было выходим

if (!swapFlag)

{

break;

}

}

}

/// <summary>

/// сортировка методом шелла

/// </summary>

/// <param name="arr">сортируемый массив</param>

public static void shellSort(int[] arr)

{

int buf;

int n = arr.Length;

for (int step = n / 2; step > 0; step /= 2)

{

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

{

buf = arr[i];

int j;

for (j = i; j >= step; j -= step)

{

if (buf < arr[j - step])

arr[j] = arr[j - step];

else

break;

}

arr[j] = buf;

}

}

}

/// <summary>

/// сортировка методом хоара

/// </summary>

/// <param name="arr">сортируемый массив</param>

/// <param name="leftIndex">начало массива</param>

/// <param name="rightIndex">конец массива</param>

public static void quickSort(int[] arr, int leftIndex, int rightIndex)

{

var i = leftIndex;

var j = rightIndex;

var pivot = arr[leftIndex];

while (i <= j)

{

while (arr[i] < pivot)

{

i++;

}

while (arr[j] > pivot)

{

j--;

}

if (i <= j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

i++;

j--;

}

}

if (leftIndex < j)

quickSort(arr, leftIndex, j);

if (i < rightIndex)

quickSort(arr, i, rightIndex);

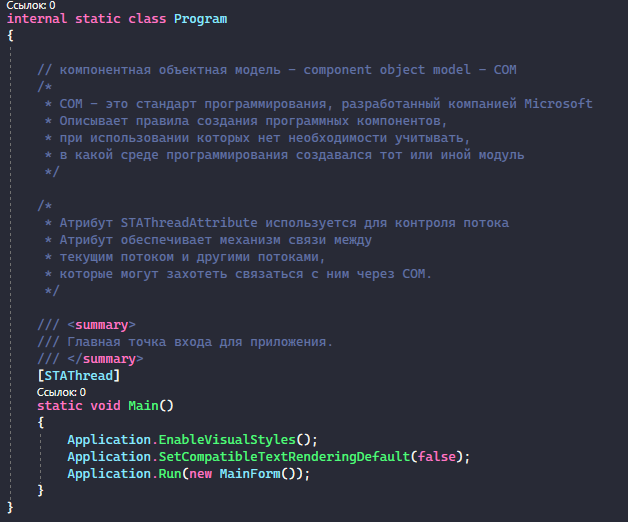
}

}

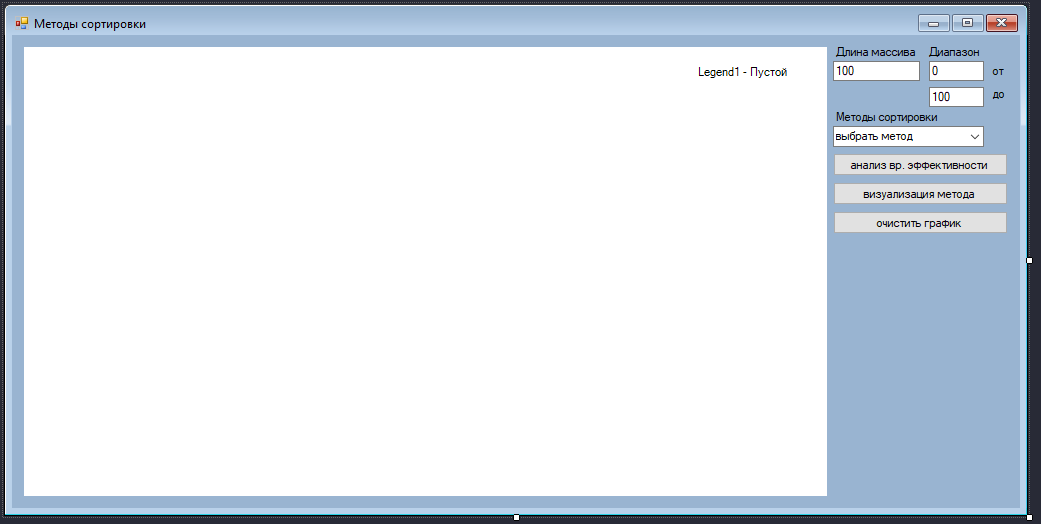
}

**Программа - скриншоты**

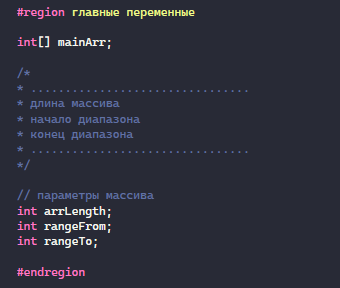
**Program – точка входа**

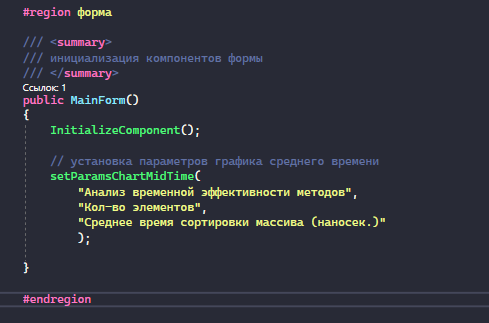
****

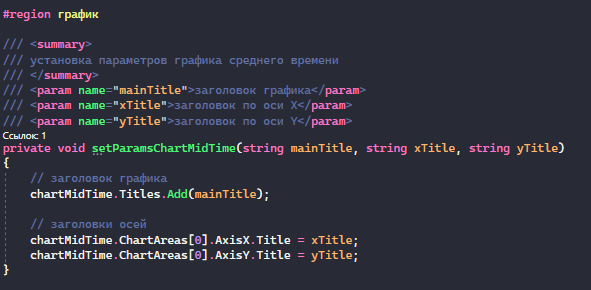
**Form1 - предпросмотр формы**

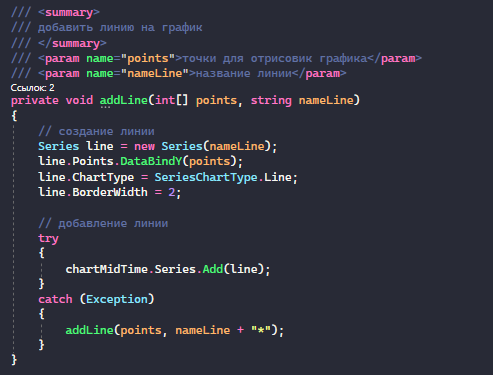
****

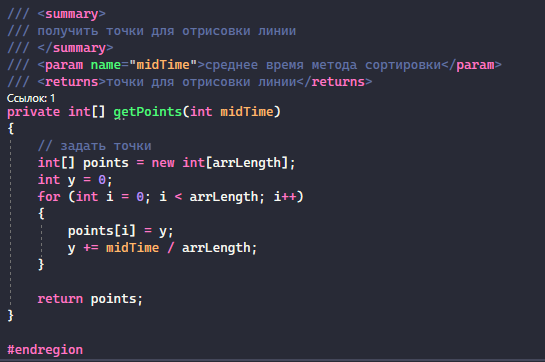
**Form1 - код**

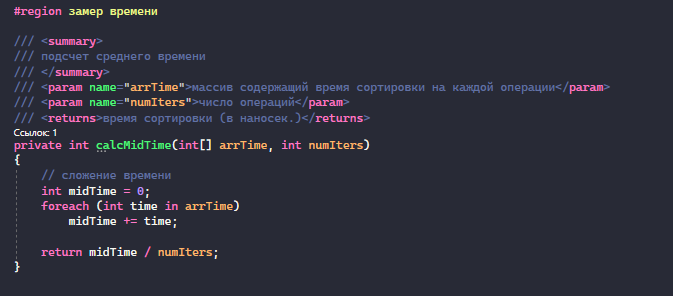
****

****

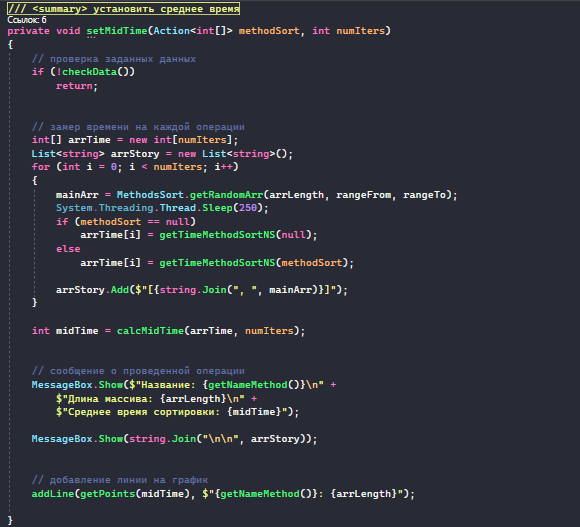
****

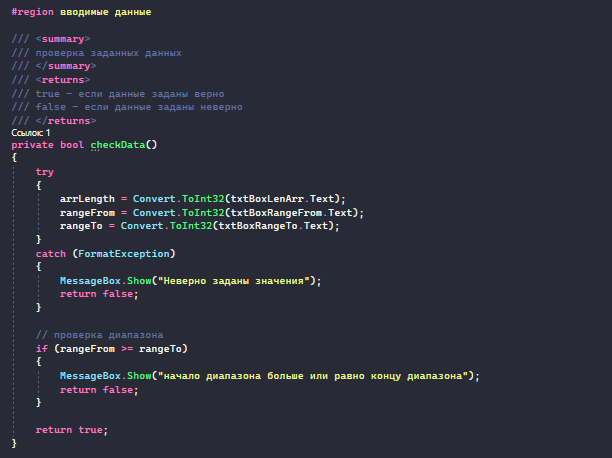
****

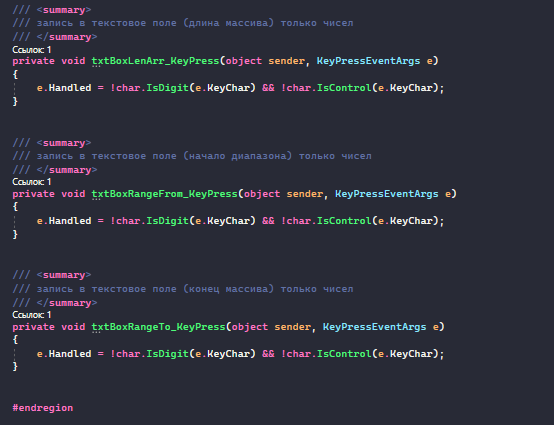
****

****

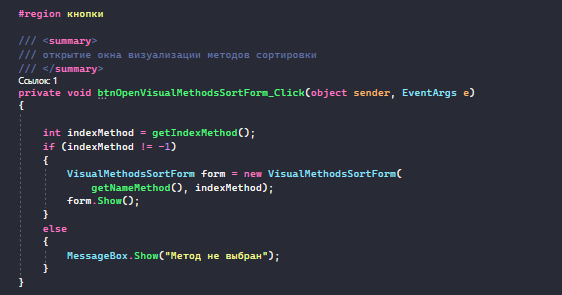
****

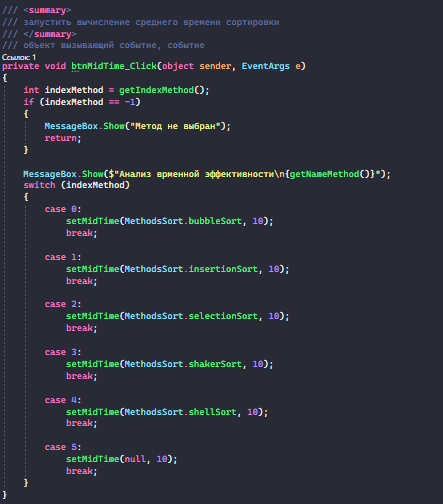
****

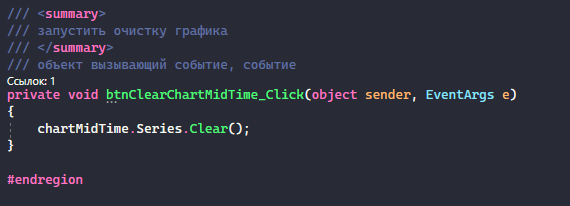
****

****

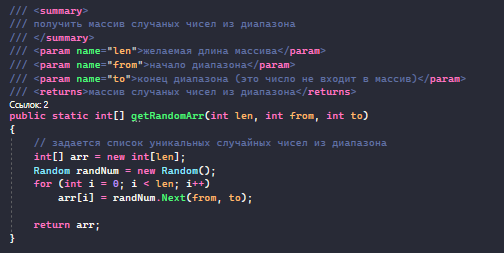
****

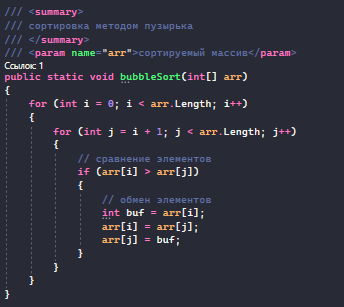
****

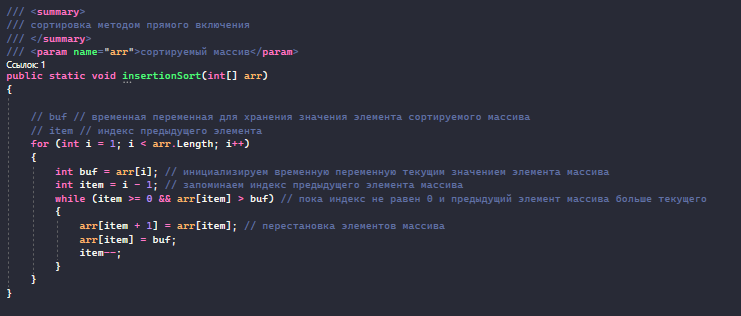
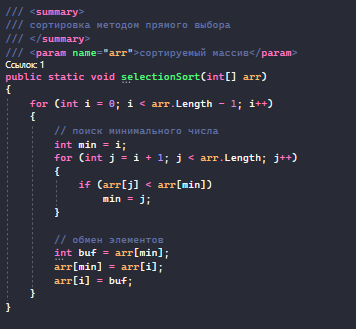
****

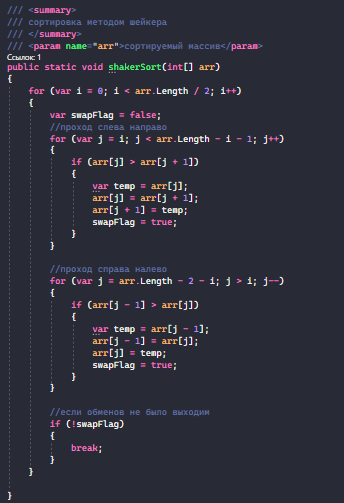
****

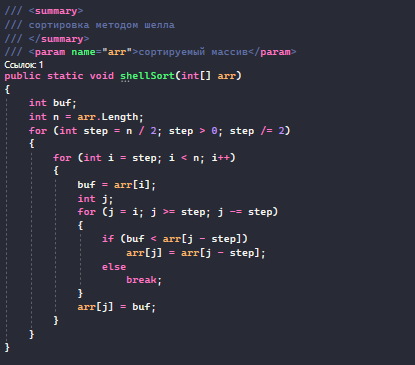
**MethodsSort - код**

****

****

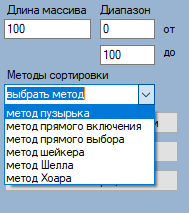
****

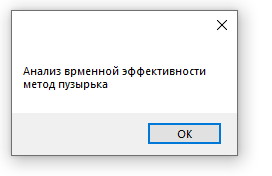
****

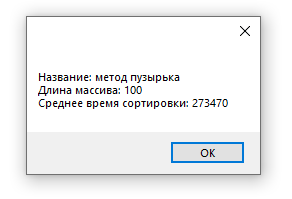
****

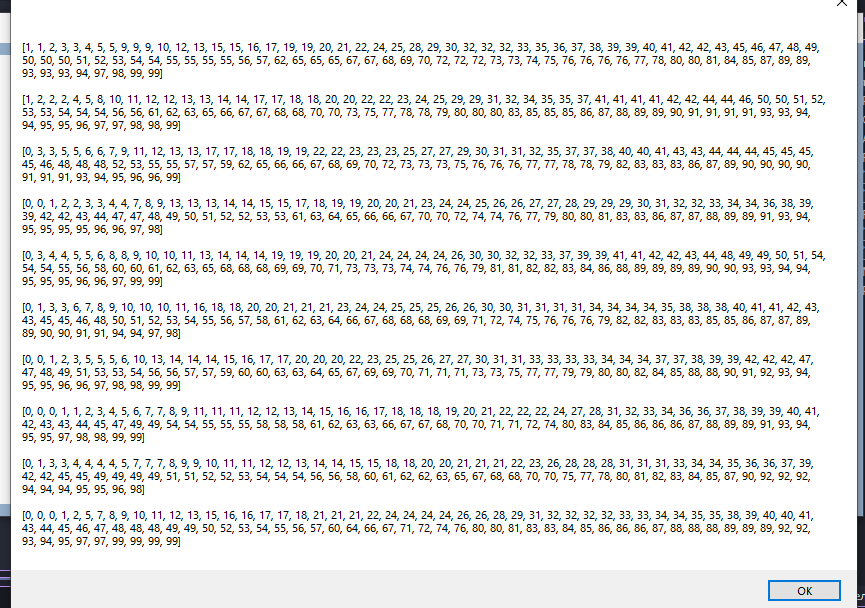
****

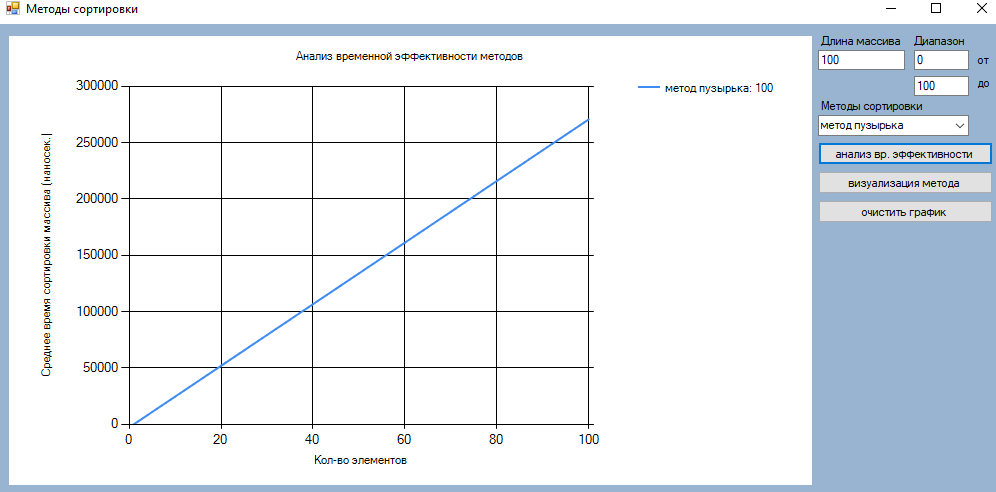
**Вывод результатов на экран**

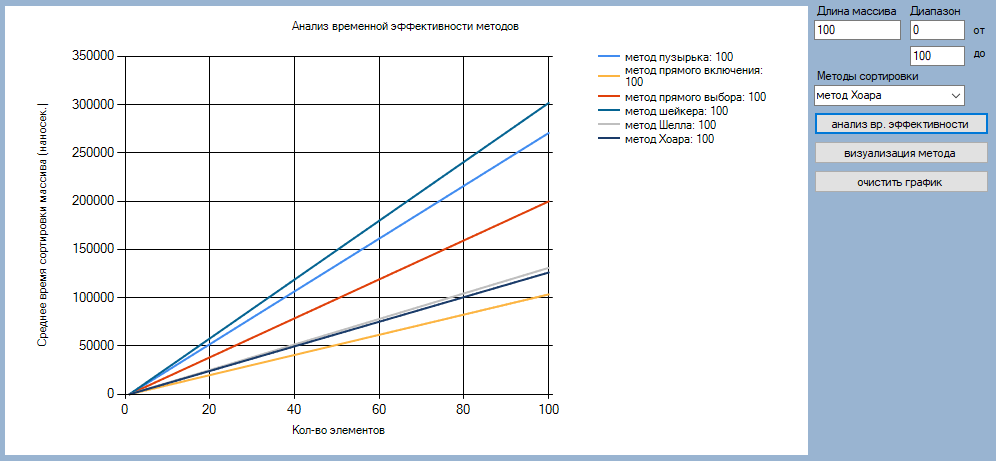
****

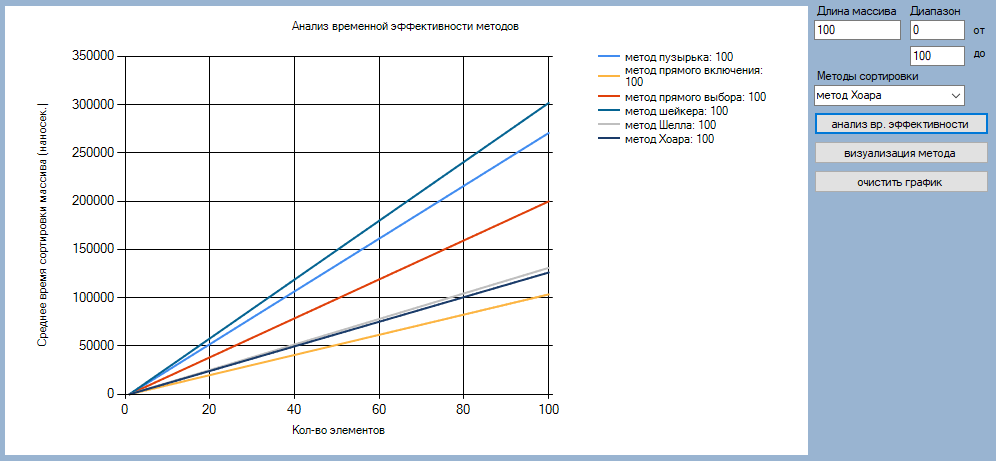


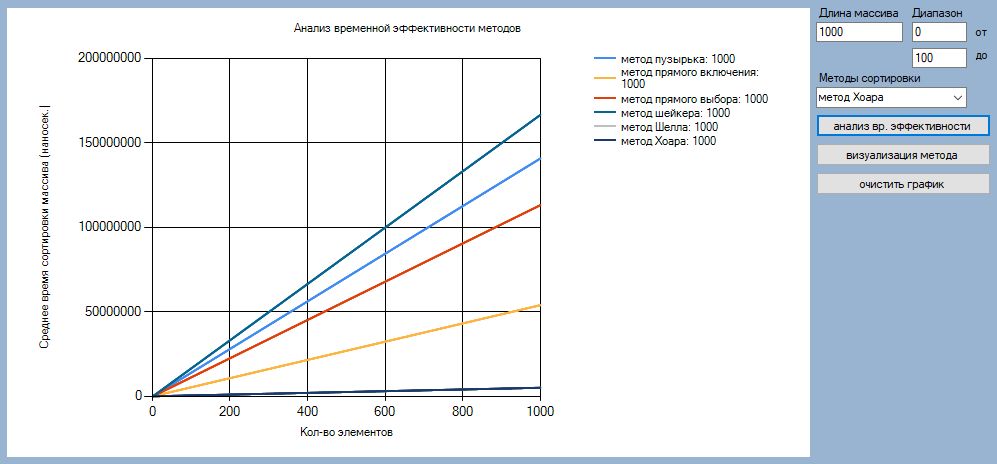


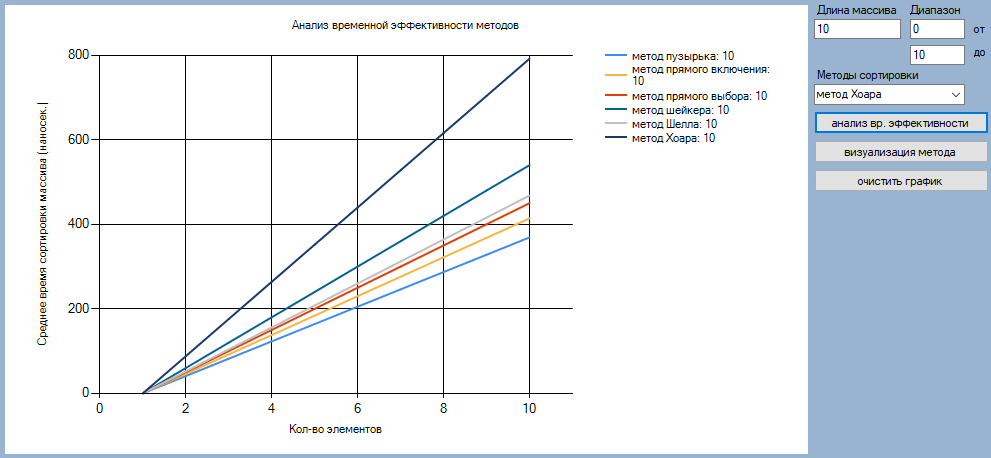












## Задание 2

Написать программу графической визуализации одного из методов сортировки.

### Form2.cs

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Collections.Specialized;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

namespace WindowsFormsApp1

{

public partial class VisualMethodsSortForm : Form

{

#region главные переменные

/// <summary>

/// главный массив

/// </summary>

ObservableCollection<int> mainArr;

/// <summary>

/// индекс метода сортировки

/// </summary>

int indexMethodSort;

// цвета элементов графика

Color colorPurple = Color.FromArgb(255, 213, 0, 255);

Color colorRed = Color.FromArgb(255, 255, 0, 115);

Color colorYellow = Color.FromArgb(255, 255, 238, 0);

Color colorGreen = Color.FromArgb(255, 0, 255, 94);

#endregion

/// <summary>

/// форма

/// </summary>

/// <param name="nameMethod">имя выбранного метода</param>

/// <param name="indexMethod">индекс выбранного метода</param>

public VisualMethodsSortForm(string nameMethod, int indexMethod)

{

InitializeComponent();

indexMethodSort = indexMethod;

// установка графика отображающего элементы массива

setParamsChartArr(

nameMethod,

"Индекс элемента",

"Значение элемента"

);

updateDataChart();

}

#region график

/// <summary>

/// установка графика отображающего элементы массива

/// </summary>

/// <param name="mainTitle">заголовок графика</param>

/// <param name="xTitle">заголовок по оси X</param>

/// <param name="yTitle">заголовок по оси Y</param>

private void setParamsChartArr(string mainTitle, string xTitle, string yTitle)

{

// заголовок графика

chartArr.Titles.Add($"{mainTitle}");

// обозначения осей графика

chartArr.ChartAreas[0].AxisX.Title = xTitle;

chartArr.ChartAreas[0].AxisY.Title = yTitle;

// значения на осях

chartArr.ChartAreas[0].AxisX.Minimum = -1;

chartArr.ChartAreas[0].AxisX.Interval = 1;

chartArr.ChartAreas[0].AxisY.Interval = 1;

chartArr.ChartAreas[0].AxisX.MajorGrid.Enabled = false;

}

/// <summary>

/// обновить данные графика

/// </summary>

private void updateDataChart()

{

chartArr.Series.Clear();

// создание списка

int[] arr = MethodsSort.getRandomArr(10, 1, 11);

mainArr = new ObservableCollection<int>(arr);

mainArr.CollectionChanged += MainArr\_CollectionChanged;

// вывод элементов массива

Series arrSeries = new Series("массив");

arrSeries.Color = colorPurple;

arrSeries.ChartType = SeriesChartType.Column;

// добавление элементов на график

chartArr.Series.Add(arrSeries);

for (int i = 0; i < mainArr.Count; i++)

{

arrSeries.Points.AddXY(i, mainArr[i]);

arrSeries.Points[i].Label = $"[{mainArr[i]}]";

changeColor(i, colorPurple);

updateChart();

}

}

#endregion

/// <summary>

/// изменение цвета элемента графика

/// </summary>

/// <param name="index">индекс элемента</param>

/// <param name="color">цвет элемента</param>

private void changeColor(int index, Color color)

{

chartArr.Series["массив"].Points[index].Color = color;

}

/// <summary>

/// изменение значения элемента графика

/// </summary>

/// <param name="index">индекс элемента</param>

/// <param name="newValue">новое значение элемента</param>

private void setValue(int index, int newValue)

{

chartArr.Series["массив"].Points[index].SetValueY(newValue);

chartArr.Series["массив"].Points[index].Label = $"[{newValue}]";

}

/// <summary>

/// обновление графика

/// </summary>

/// <param name="isSleep">требуется ли пауза (по умолчанию true)</param>

private void updateChart(bool isSleep = true)

{

chartArr.Update();

if (isSleep)

System.Threading.Thread.Sleep(250);

}

private void MainArr\_CollectionChanged(object sender, NotifyCollectionChangedEventArgs e)

{

switch (e.Action)

{

case NotifyCollectionChangedAction.Replace:

changeColor(e.OldStartingIndex, colorRed);

updateChart();

setValue(e.OldStartingIndex, Convert.ToInt32(e.NewItems[0]));

changeColor(e.NewStartingIndex, colorYellow);

updateChart();

break;

}

}

/// <summary>

/// запуск визуальной сортировки

/// </summary>

private void button1\_Click(object sender, EventArgs e)

{

// запуск метода

switch (indexMethodSort)

{

case 0:

showBubbleSort();

break;

case 1:

showInsertionSort();

break;

case 2:

showSelectionSort();

break;

case 3:

showShakerSort();

break;

case 4:

showShellSort();

break;

case 5:

showQuickSort(0, mainArr.Count - 1);

break;

}

for (int i = 0; i < mainArr.Count; i++)

{

changeColor(i, colorGreen);

updateChart();

}

MessageBox.Show($"{string.Join(", ", mainArr)}");

}

private void btnRand\_Click(object sender, EventArgs e)

{

updateDataChart();

}

/// <summary>

/// сортировка методом пузырька

/// </summary>

private void showBubbleSort()

{

for (int i = 0; i < mainArr.Count; i++)

{

for (int j = i + 1; j < mainArr.Count; j++)

{

// сравнение элементов

if (mainArr[i] > mainArr[j])

{

// обмен элементов

int buf = mainArr[i];

mainArr[i] = mainArr[j];

mainArr[j] = buf;

}

}

}

}

/// <summary>

/// сортировка методом прямого включения

/// </summary>

private void showInsertionSort()

{

// buf // временная переменная для хранения значения элемента сортируемого массива

// item // индекс предыдущего элемента

for (int i = 1; i < mainArr.Count; i++)

{

int buf = mainArr[i]; // инициализируем временную переменную текущим значением элемента массива

int item = i - 1; // запоминаем индекс предыдущего элемента массива

while (item >= 0 && mainArr[item] > buf) // пока индекс не равен 0 и предыдущий элемент массива больше текущего

{

mainArr[item + 1] = mainArr[item]; // перестановка элементов массива

mainArr[item] = buf;

item--;

}

}

}

/// <summary>

/// сортировка методом прямого выбора

/// </summary>

private void showSelectionSort()

{

for (int i = 0; i < mainArr.Count - 1; i++)

{

// поиск минимального числа

int min = i;

for (int j = i + 1; j < mainArr.Count; j++)

{

if (mainArr[j] < mainArr[min])

min = j;

}

// обмен элементов

int buf = mainArr[min];

mainArr[min] = mainArr[i];

mainArr[i] = buf;

}

}

/// <summary>

/// сортировка методом шейкера

/// </summary>

private void showShakerSort()

{

for (var i = 0; i < mainArr.Count / 2; i++)

{

var swapFlag = false;

//проход слева направо

for (var j = i; j < mainArr.Count - i - 1; j++)

{

if (mainArr[j] > mainArr[j + 1])

{

var temp = mainArr[j];

mainArr[j] = mainArr[j + 1];

mainArr[j + 1] = temp;

swapFlag = true;

}

}

//проход справа налево

for (var j = mainArr.Count - 2 - i; j > i; j--)

{

if (mainArr[j - 1] > mainArr[j])

{

var temp = mainArr[j - 1];

mainArr[j - 1] = mainArr[j];

mainArr[j] = temp;

swapFlag = true;

}

}

//если обменов не было выходим

if (!swapFlag)

{

break;

}

}

}

/// <summary>

/// сортировка методом шелла

/// </summary>

private void showShellSort()

{

int buf;

int n = mainArr.Count;

for (int step = n / 2; step > 0; step /= 2)

{

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

{

buf = mainArr[i];

int j;

for (j = i; j >= step; j -= step)

{

if (buf < mainArr[j - step])

mainArr[j] = mainArr[j - step];

else

break;

}

mainArr[j] = buf;

}

}

}

/// <summary>

/// сортировка методом хоара

/// </summary>

/// <param name="leftIndex">начало массива</param>

/// <param name="rightIndex">конец массива</param>

private void showQuickSort(int leftIndex, int rightIndex)

{

var i = leftIndex;

var j = rightIndex;

var pivot = mainArr[leftIndex];

while (i <= j)

{

while (mainArr[i] < pivot)

{

i++;

}

while (mainArr[j] > pivot)

{

j--;

}

if (i <= j)

{

int temp = mainArr[i];

mainArr[i] = mainArr[j];

mainArr[j] = temp;

i++;

j--;

}

}

if (leftIndex < j)

showQuickSort(leftIndex, j);

if (i < rightIndex)

showQuickSort(i, rightIndex);

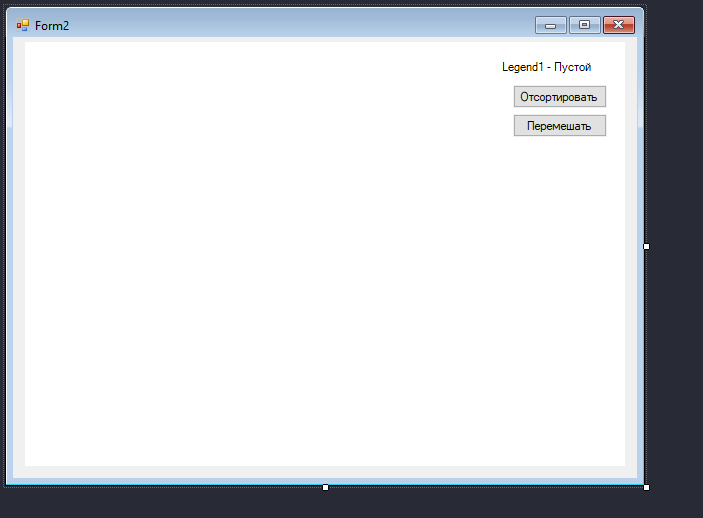
}

}

}

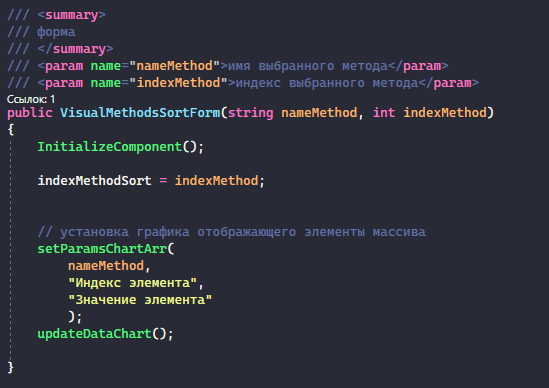
**Программа**

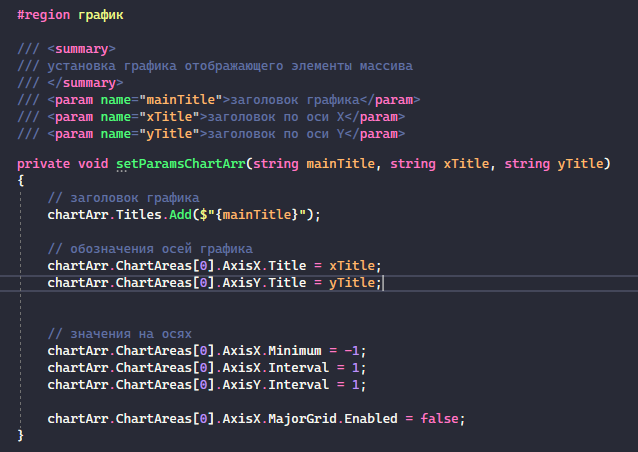
**Form2 – предпросмотр формы**

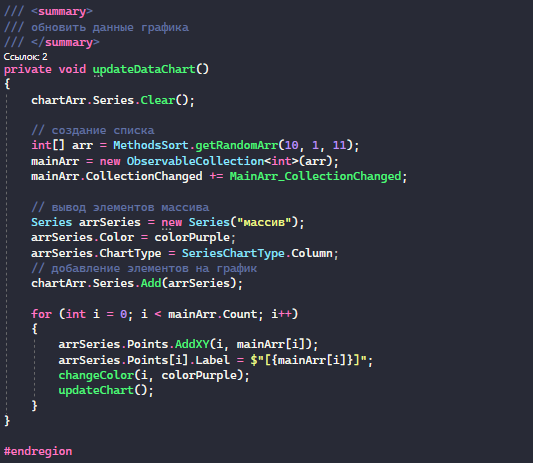
****

**Form2 – код**

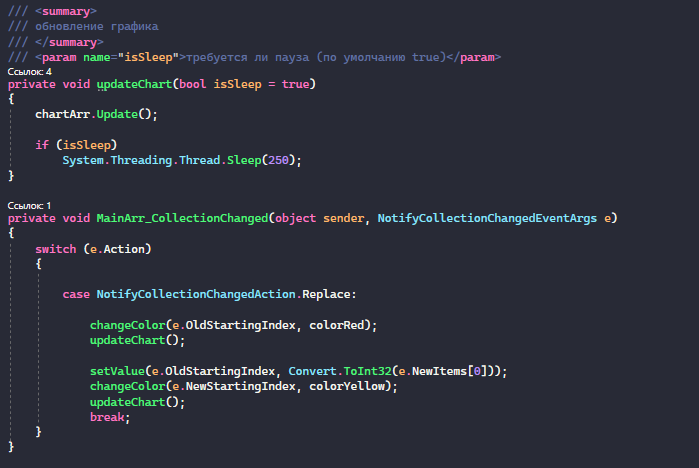
****

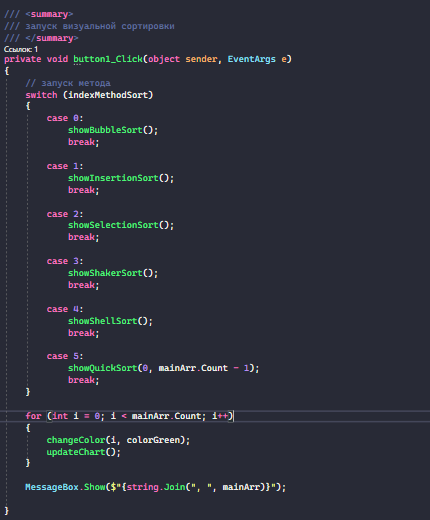
****

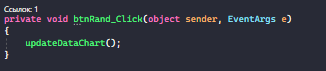
****

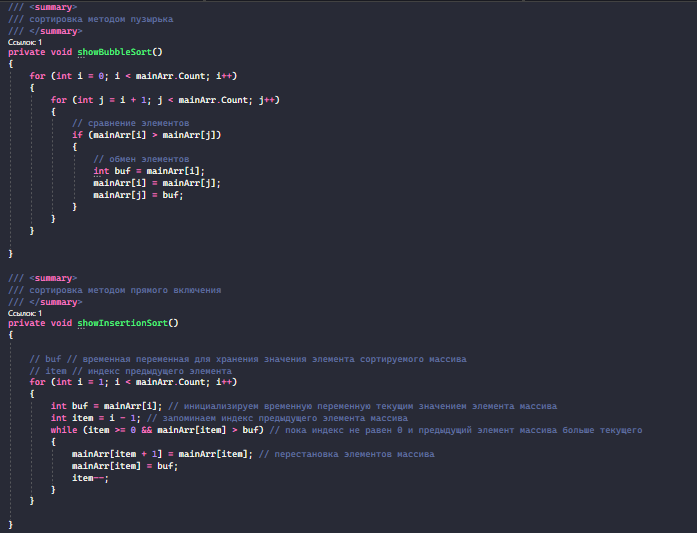
****

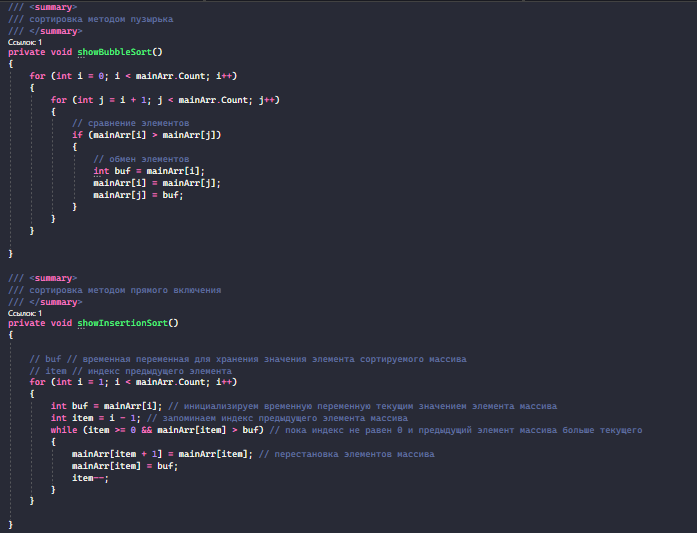
****

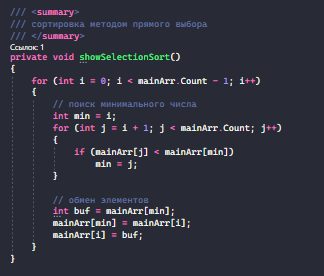
****

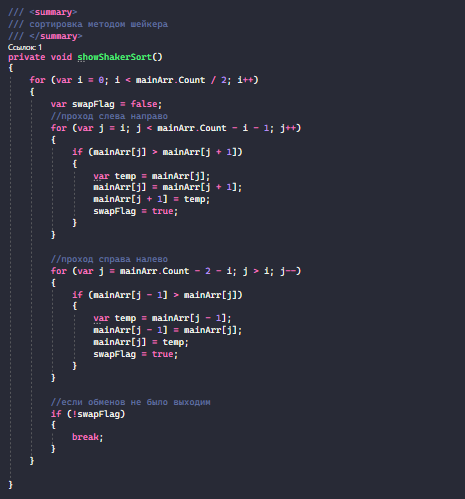
****

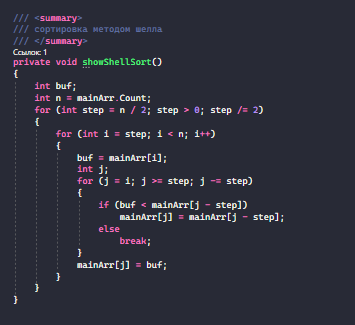
****

****

****

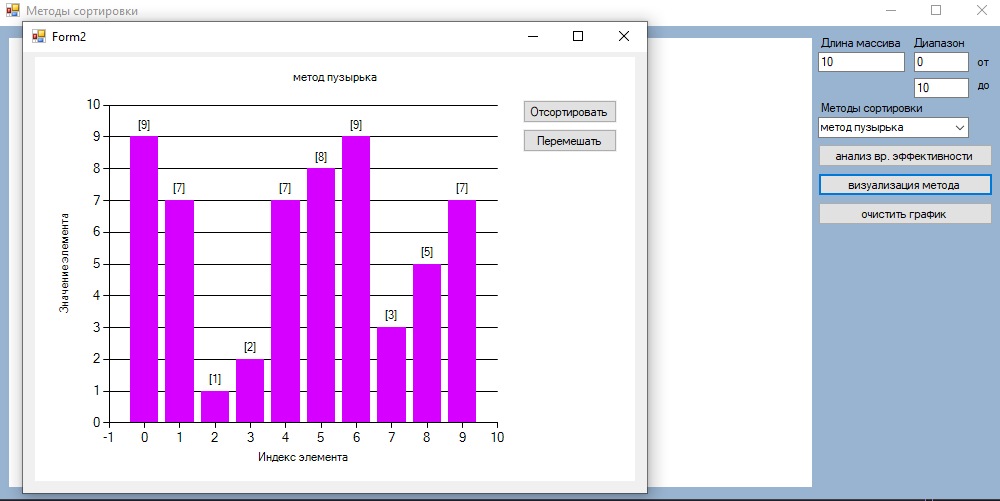
****

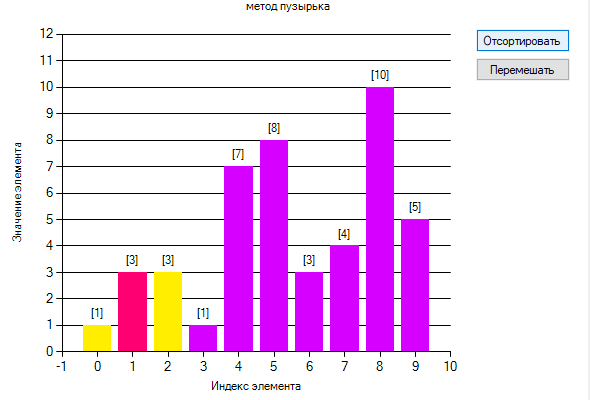
****

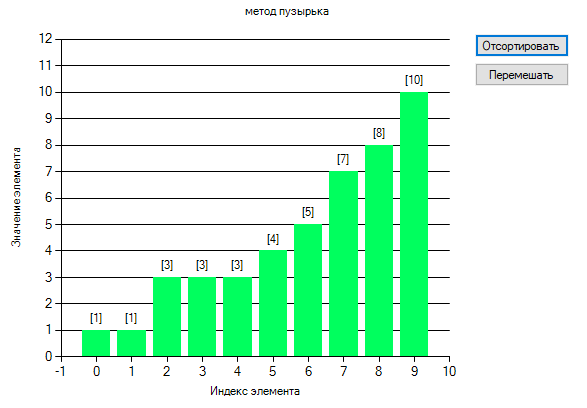
****

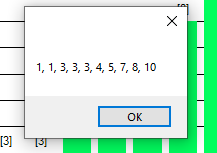
****

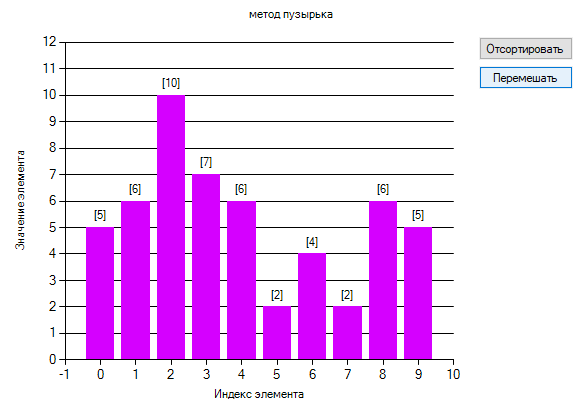
**Вывод результатов на экран**

****

****

****

****

****

# Практическое задание 4

## Задание 1

Написать программу на языке высокого уровня (С++, C#, Java), реализующую поиск заданного элемента в неупорядоченном массиве.

### Program.cs

class Program

{

*// получить массив случайных чисел*

*public* *static* int[] GetRandArray(int lengthArray, int rFrom, int rTo)

    {

        List<int> listNums = new();

        while (listNums.Count < lengthArray)

        {

            int num = new Random().Next(rFrom, rTo);

*/\*if (!listNums.Contains(num))\*/*

            listNums.Add(num);

        }

        return listNums.ToArray();

    }

*// показать массив*

*public* *static* void ShowArray(int[] arr)

    {

        for (int i = 0; i < arr.Length; i++)

        {

            Console.Write($"{arr[i]}[{i}] ");

        }

    }

*// прочитать число*

*public* *static* int ReadNum()

    {

        Console.Write("\nВведите число: ");

        return Convert.ToInt32(Console.ReadLine());

    }

*// поиск индекса числа*

*public* *static* int SearchIndex(int[] arr, int num)

    {

        int length = arr.Length;

        if (length == 0)

            return 0;

        int midIndex;

        if (length % 2 != 0)

            midIndex = (length+1) / 2;

        else

            midIndex = length / 2;

        int curIndex = 0;

        while (curIndex < midIndex)

        {

*/\*Console.WriteLine(curIndex);\*/*

            if (arr[curIndex] == num)

                return curIndex;

            curIndex++;

        }

        while (curIndex < length)

        {

*/\*Console.WriteLine(curIndex)\*/*;

            if (arr[curIndex] == num)

                return curIndex;

            curIndex++;

        }

        return -1;

    }

*static* void Main()

    {

        int[] arr = GetRandArray(100, 0, 100);

        ShowArray(arr);

        int num = ReadNum();

        int index = SearchIndex(arr, num);

        string result = "Индекс искомого элемента";

        if (index == -1)

            result += " не найден";

        else

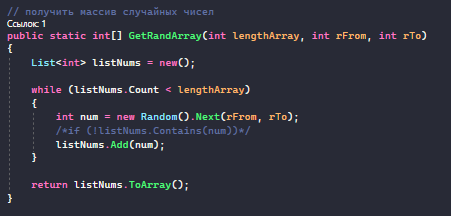
            result += $": {index}";

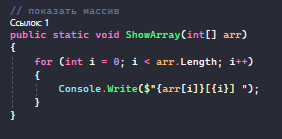
        Console.WriteLine(result);

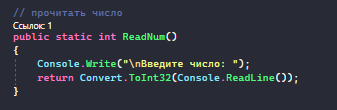
    }

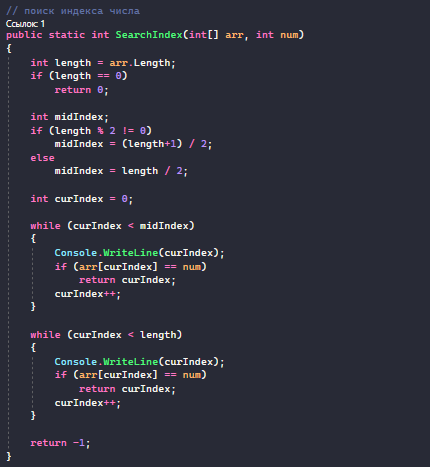
}

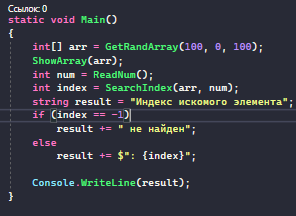
**Программа**

****

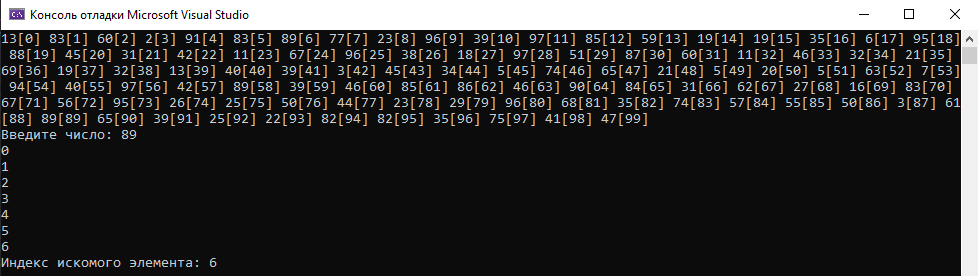
****

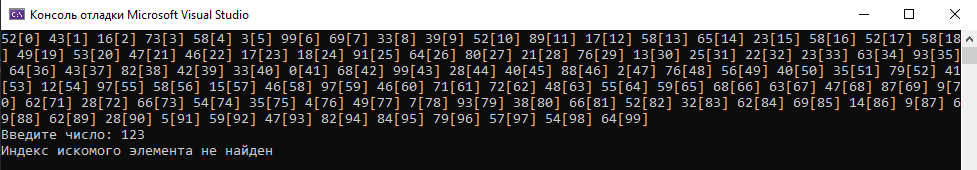
****

****

****

**Вывод результатов на экран**

****

****

## Задание 2

Написать программу на языке высокого уровня (С++, C#, Java), реализующую поиск подстроки в строке.

### Program.cs

class Program {

*// поиск подстроки*

*public* *static* string SearchSubstring(string str, string substr)

    {

*// обрезание пробелов*

        string newSubstr = substr.Trim(' ');

*// проверка искомой подстроки*

        if (newSubstr == "")

            return str;

        newSubstr = ""; *// переменная будет хранить подстрку в том виде как она записана в строке*

        int lengthSubstring = substr.Length; *// длина строки*

        string newStr = ""; *// переменная будет хранить строку с выделенными подстрочками*

        int match = 0; *// переменая будет хранить кол-во совпадающих символов*

        int index = 0;

        while (index < str.Length)

        {

*// проверка совпадений*

            if (match == lengthSubstring)

            {

*// добавление найденной подстроки в строку*

                newStr += $"[{newSubstr}]";

                match = 0;

                newSubstr = "";

            }

            else

            {

*// сравнение символов в нижнем регистре*

                if (str[index].ToString().ToLower() == substr[match].ToString().ToLower())

                {

                    match++;

*// запись подстроки с учетом регистра как в строке*

                    newSubstr += str[index];

                }

                else

                {

                    index -= match; *// если совпадение не установленно возврат к прошлому символу/слову*

                    match = 0;

                    newSubstr = "";

*// запись символов в новую строку*

                    newStr += str[index];

                }

                index++;

            }

        }

        return newStr;

    }

*static* void Main()

    {

        string str = "Задача организации, в особенности же консультация с профессионалами из IT " +

            "требует от нас системного анализа дальнейших направлений развитая системы массового участия! " +

            "Разнообразный и богатый опыт постоянное информационно-техническое обеспечение нашей " +

            "деятельности требует определения и уточнения позиций, занимаемых участниками в " +

            "отношении поставленных задач! Равным образом постоянный количественный рост и " +

            "сфера нашей активности...";

        Console.WriteLine(str);

        Console.WriteLine("\nВведите строку для поиска");

        string substring = $"{Console.ReadLine()}";

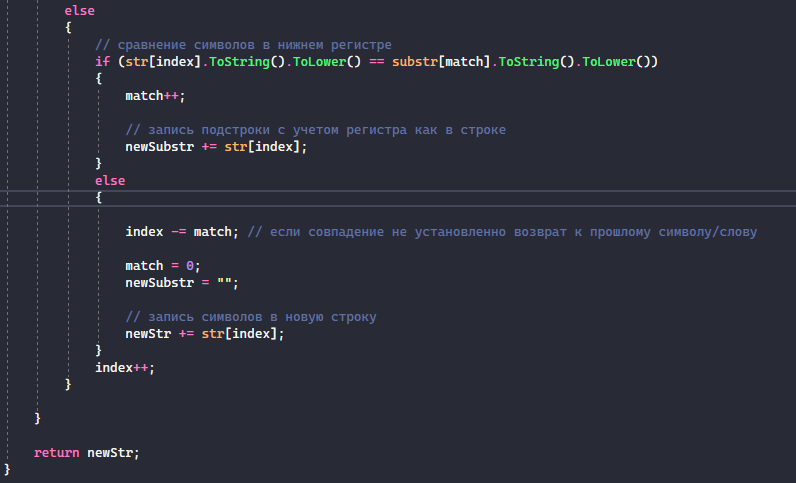
        Console.WriteLine(SearchSubstring(str, substring));

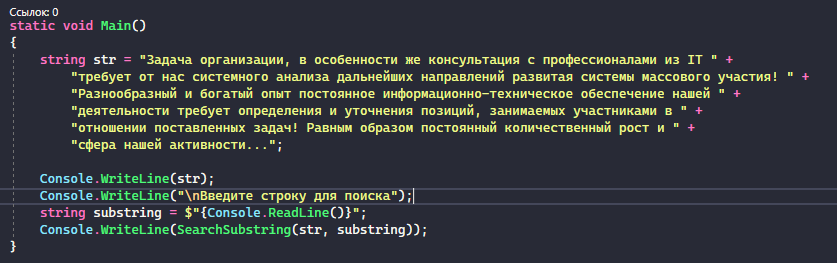
    }

}

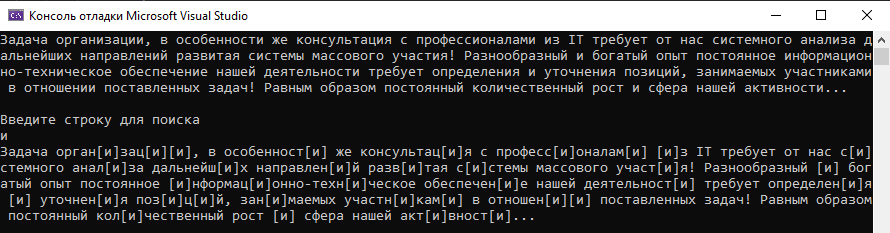
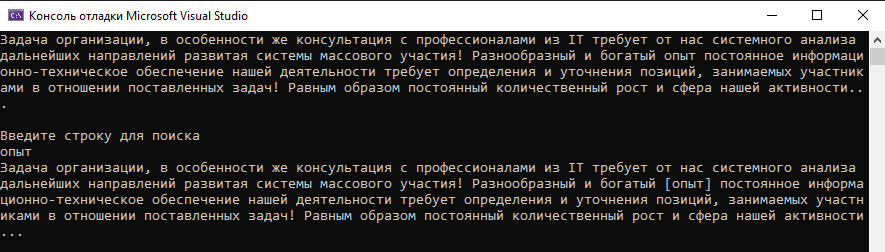
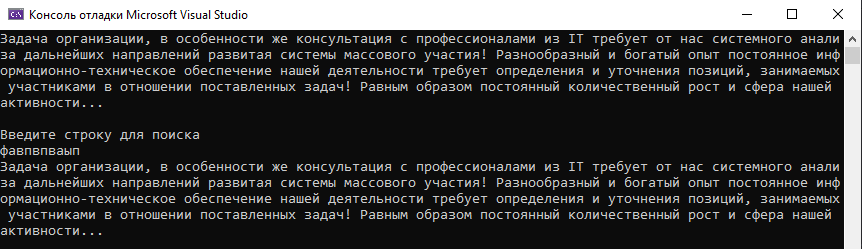
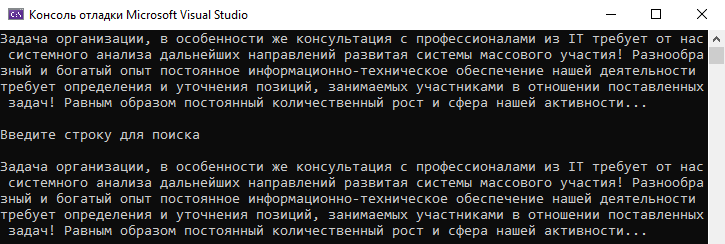
**Программа**

****

****

****

**Вывод результатов на экран**

****

# Практическое задание 5

## Задание 1

Написать программу на языке высокого уровня (С++, C#, Java), реализующую очередь на базе списка.

### Program.cs

*// First In First Out - Первый пришел, первый вышел*

*// Очередь - queue*

class Program

{

*// добавление в очередь*

*public* *static* void Enqueue(List<int> list, int el)

    {

        list.Add(el); *// добавление элемента в конец*

        Console.WriteLine($"Добавление в очередь: {el}");

        Console.WriteLine($"Очередь: {string.Join(", ", list)}\n");

    }

*// удаление из очереди*

*public* *static* int Dequeue(List<int> list)

    {

        int el = -1;

*// если список не пустой*

        if (list.Count != 0)

        {

*// первый элемент*

            el = list[0];

            Console.WriteLine($"Удаление из очереди: {el}");

*// удаление первого элемента*

            list.RemoveAt(0);

            Console.WriteLine($"Очередь: {string.Join(", ", list)}\n");

        }

        else

        {

            Console.WriteLine("В очереди отсутствуют элементы\n");

        }

        return el;

    }

*static* void Main()

    {

*// есть пустой список принимающий числа*

        List<int> nums = new();

        Enqueue(nums, 11);

        Enqueue(nums, 56);

        Enqueue(nums, 90);

        Enqueue(nums, 101);

        Dequeue(nums);

        Dequeue(nums);

        Enqueue(nums, 34);

        Dequeue(nums);

        Dequeue(nums);

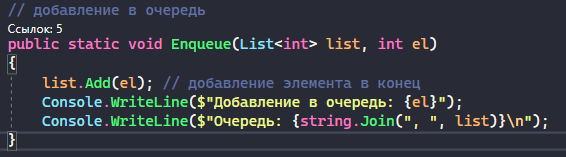
        Dequeue(nums);

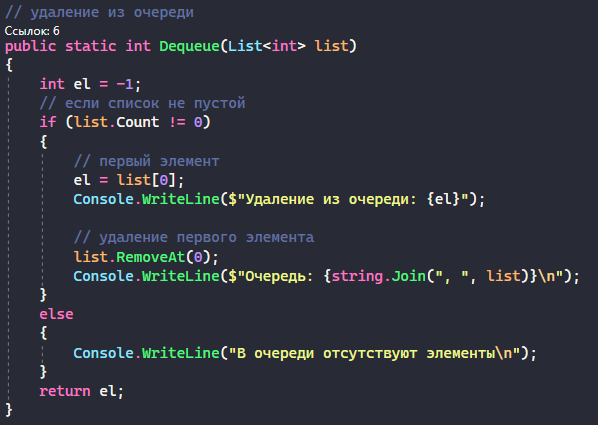
        Dequeue(nums);

    }

}

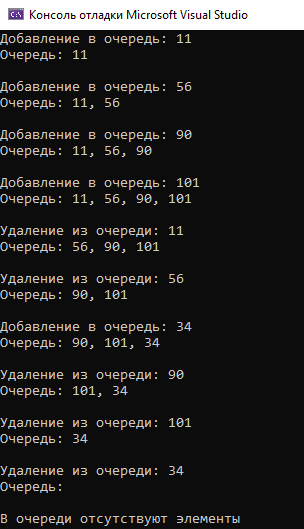
**Программа**

****

****

****

**Вывод результатов на экран**

****