

Міністерство освіти України
Національний технічний університет "ХПІ"
кафедра "Інформатики та інтелектуальної власності"

Звіт
Лабораторна робота 2
з дисципліни "DOT NET"

Виконав: студент групи КН-921Б
Бірюков Д. Є.
Перевірив:
Івашко А.В.

ЗМІСТ

Завдання 1	3
Завдання 2	6
Завдання 3	8
Завдання 4	10
Завдання 5	13
Завдання 6	15
Завдання 7	17
Завдання 8	19
Завдання 9	22
Висновок	25

Завдання 1

Використовуючи завдання власного варіанта до лабораторної роботи 8 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

Объявить массив целых чисел и заполнить его случайными значениями. Размер массива и диапазон значений его элементов заданы в Вашем варианте индивидуального задания. В индивидуальных заданиях указано также, какую обработку массива следует произвести.

Для всех вариантов задания следует иметь в виду следующее:

- 0 считается положительным числом, если в задании не оговорен какой-то другой его статус;
- когда речь идет о какой-то последовательности чисел, имеется в виду последовательность с длиной, большей 1;
- в тех случаях, когда задание требует выполнения каких-то вычислений, разрешается выполнять их с той точностью, которую обеспечивают операции целочисленной арифметики.

№ варианта	Размерность массива	Диапазон значений	Что нужно сделать
2	200	-50 - 50	Подсчитать количество пар соседних элементов с одинаковыми значениями

using System;

class Program

```
{
    static void Main()
    {
        int[] Ar = new int[100]; // массив, который обрабатывается

        Random random = new Random(); // создаем генератор случайных чисел

        // Заполнение массива случайными числами
        for (int i = 0; i < 100; i++)
        {
            Ar[i] = random.Next(-50, 51); // диапазон значений от -50 до 50
        }

        // Вывод начального массива
        Console.WriteLine("Начальный массив:");
        for (int i = 0; i < 100; i++)
        {
            Console.Write($"{Ar[i],3} ");
        }
        Console.WriteLine();
        Console.WriteLine();

        int nn = 0; // количество элементов в последовательности
        int ib = 0; // индекс начала последовательности
        int av = 0; // среднее значение
```

```

int pairCount = 0; // количество пар соседних элементов с одинаковыми значениями

for (int i = 0; i < 100; i++)
{
    if (Ar[i] < 0)
    {
        // Обработка отрицательного элемента
        if (nn == 0)
        {
            // Начало последовательности
            ib = i;
            av = Ar[i];
            nn = 1;
        }
        else
        {
            // Накопление суммы и подсчет количества
            av += Ar[i];
            nn++;
        }
    }
    else
    {
        // Обработка положительного элемента
        if (nn > 0)
        {
            // Если есть необработанная отрицательная последовательность
            av /= nn; // Усреднение

            // Подсчет пар соседних элементов с одинаковыми значениями
            for (int j = ib; j < i; j++)
            {
                if (Ar[j] > av)
                {
                    Ar[j] = av;
                }
                else if (j > 0 && Ar[j] == Ar[j - 1])
                {
                    pairCount++;
                }
            }

            nn = 0; // Последовательность обработана
        }
    }
}

if (nn > 0)
{
    // Если не обработана последняя отрицательная последовательность
    av /= nn;
}

```

```

    for (int j = ib; j < 100; j++)
    {
        if (Ar[j] > av)
        {
            Ar[j] = av;
        }
        else if (j > 0 && Ar[j] == Ar[j - 1])
        {
            pairCount++;
        }
    }
}

// Вывод результатов
Console.WriteLine("Массив-результат:");
for (int i = 0; i < 100; i++)
{
    Console.Write($"{Ar[i],3} ");
}
Console.WriteLine();

// Вывод количества пар соседних элементов с одинаковыми значениями
Console.WriteLine($"Количество пар соседних элементов с одинаковыми значениями:
    {pairCount}");
}
}

```

```

Массив-результат:
16 31 29 20 30 9 -27 42 -48 4 -49 3 30 -24 -12 -12 9 39 28 16 -22 -35 6 -31 -31
10 16 -41 -48 -48 7 35 -50 -44 50 -31 -20 24 -48 -47 29 22 -42 -50 -37 6 -11 32 -36 41
-39 -45 -38 -32 22 -35 34 48 37 2 -44 30 18 23 17 44 12 31 -34 -42 -43 0 -42 -41 -35
-42 -48 -35 33 9 4 -49 -29 0 -31 0 34 -24 -16 -14 -15 24 -41 2 31 -18 -26 0 -16 21
Количество пар соседних элементов с одинаковыми значениями: 2

C:\Users\dmitriybirukov\source\repos\Lab_02\Task_01\bin\Debug\net6.0\Task_01.exe (процесс 12316) за-
вершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры
" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно:

```

- Завдання 2*
Виконати минуле завдання, зробивши похідний масив динамічним, задавши його розмір з консолі.

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        Console.WriteLine("Введите размер массива: ");
        if (!int.TryParse(Console.ReadLine(), out int arraySize) || arraySize <= 0)
        {
            Console.WriteLine("Некорректный ввод размера массива.");
            return;
        }

        int[] Ar = new int[arraySize]; // динамический массив, который обрабатывается

        Random random = new Random(); // создаем генератор случайных чисел

        // Заполнение массива случайными числами
        for (int i = 0; i < arraySize; i++)
        {
            Ar[i] = random.Next(-50, 51); // диапазон значений от -50 до 50
        }

        // Вывод начального массива
        Console.WriteLine("Начальный массив:");
        for (int i = 0; i < arraySize; i++)
        {
            Console.WriteLine($"{Ar[i],3} ");
        }
        Console.WriteLine();
        Console.WriteLine();

        int nn = 0; // количество элементов в последовательности
        int ib = 0; // индекс начала последовательности
        int av = 0; // среднее значение

        for (int i = 0; i < arraySize; i++)
        {
            if (Ar[i] < 0)
            {
                // Обработка отрицательного элемента
                if (nn == 0)
                {
                    // Начало последовательности
                    ib = i;
                    av = Ar[i];
                    nn = 1;
                }
            }
            else
            {
                // Накопление суммы и подсчет количества
                av += Ar[i];
                nn++;
            }
        }
        else
        {
            // Обработка положительного элемента

```

```

        if (nn > 0)
        {
            // Если есть необработанная отрицательная последовательность
            av /= nn; // Усреднение

            // Подсчет пар соседних элементов с одинаковыми значениями
            for (int j = ib; j < i; j++)
            {
                if (Ar[j] > av)
                {
                    Ar[j] = av;
                }
            }

            nn = 0; // Последовательность обработана
        }
    }
}

if (nn > 0)
{
    // Если не обработана последняя отрицательная последовательность
    av /= nn;

    for (int j = ib; j < arraySize; j++)
    {
        if (Ar[j] > av)
        {
            Ar[j] = av;
        }
    }
}

// Вывод результатов
Console.WriteLine("Массив-результат:");
for (int i = 0; i < arraySize; i++)
{
    Console.Write($" {Ar[i],3} ");
}
Console.WriteLine();

// Подсчет количества пар соседних элементов с одинаковыми значениями
int pairCount = Enumerable.Range(1, arraySize - 1).Count(i => Ar[i] == Ar[i - 1]);
Console.WriteLine($"Количество пар соседних элементов с одинаковыми значениями: {pairCount}");
}
}

```

```

Консоль отладки Microsoft V
Введите размер массива: 200
Начальный массив:
-14 23 7 -31 -46 -41 -35 -27 -18 -21 46 -5 16 13 32 46 -20 -24 -18 -33 -31 7 44 -3 -47 -3 -20 15 14 4
31 16 -5 43 -28 3 -48 -46 -7 10 44 -15 22 42 30 -49 -24 -1 -1 41 49 -18 -19 -29 -5 40 -50 -2 -29 19
44 13 10 23 36 5 -38 2 14 -21 7 -34 -34 33 -40 -18 8 -28 37 4 -37 -47 43 -26 22 49 -16 47 -23 35
46 6 -50 -4 -26 -29 44 2 18 32 15 -22 -6 4 -6 21 45 -22 14 28 -19 48 18 -36 17 -18 40 -22 -25 48
49 39 -11 7 -18 -26 24 4 31 -8 38 6 -6 36 -11 -13 37 23 38 -39 48 1 -9 46 -23 -2 27 18 -28 9
13 -17 43 46 38 12 24 44 42 -13 -24 6 46 34 43 28 9 42 -18 12 0 32 -33 -35 -34 8 47 46 27 -13
-4 -46 -18 -47 -35 -7 -49 -39 35 -46 23 41 -18 29 3 -11 -9 -7 -8 49

Массив-результат:
-14 23 7 -31 -46 -41 -35 -31 -31 -31 46 -5 16 13 32 46 -25 -25 -25 -33 -31 7 44 -18 -47 -18 -20 15 14 4
31 16 -5 43 -28 3 -48 -46 -33 10 44 -15 22 42 30 -49 -24 -18 -18 41 49 -18 -19 -29 -17 40 -50 -27 -29 19
44 13 10 23 36 5 -38 2 14 -21 7 -34 -34 33 -40 -29 8 -28 37 4 -42 -47 43 -26 22 49 -16 47 -23 35
46 6 -50 -27 -27 -29 44 2 18 32 15 -22 -14 4 -6 21 45 -22 14 28 -19 48 18 -36 17 -18 40 -23 -25 48
49 39 -11 7 -22 -26 24 4 31 -8 38 6 -6 36 -12 -13 37 23 38 -39 48 1 -9 46 -23 -12 27 18 -28 9
13 -17 43 46 38 12 24 44 42 -18 -24 6 46 34 43 28 9 42 -18 12 0 32 -34 -35 -34 8 47 46 27 -28
-28 -46 -28 -47 -35 -28 -49 -39 35 -46 23 41 -18 29 3 -11 -9 -8 -8 49

Количество пар соседних элементов с одинаковыми значениями: 9

```

- Завдання 3*

Виконати минуле завдання, використовуючи кольорові можливості консолі для покращення інтерфейсу роботи програми. Наприклад, виділяти різними кольорами знайдені в масивах за завданням елементи чи їхні послідовності, розраховані параметри, тощо.

```
using System;
using System.Linq;

class Program
{
    static void Main()
    {
        Console.Write("Введите размер массива: ");
        if (!int.TryParse(Console.ReadLine(), out int arraySize) || arraySize <= 0)
        {
            Console.WriteLine("Некорректный ввод размера массива.");
            return;
        }

        int[] Ar = new int[arraySize]; // динамический массив, который обрабатывается

        Random random = new Random(); // создаем генератор случайных чисел

        // Заполнение массива случайными числами
        for (int i = 0; i < arraySize; i++)
        {
            Ar[i] = random.Next(-50, 51); // диапазон значений от -50 до 50
        }

        // Вывод начального массива с цветом
        Console.WriteLine("Начальный массив:");
        for (int i = 0; i < arraySize; i++)
        {
            Console.Write($"{Ar[i],3} ");
        }
        Console.WriteLine();
        Console.WriteLine();

        int nn = 0; // количество элементов в последовательности
        int ib = 0; // индекс начала последовательности
        int av = 0; // среднее значение

        for (int i = 0; i < arraySize; i++)
        {
            if (Ar[i] < 0)
            {
                // Обработка отрицательного элемента
                if (nn == 0)
                {
                    // Начало последовательности
                    ib = i;
                    av = Ar[i];
                }
            }
        }
    }
}
```



```

        nn = 1;
    }
    else
    {
        // Накопление суммы и подсчет количества
        av += Ar[i];
        nn++;
    }
}
else
{
    // Обработка положительного элемента
    if (nn > 0)
    {
        // Если есть необработанная отрицательная последовательность
        av /= nn; // Усреднение

        // Подсчет пар соседних элементов с одинаковыми значениями
        for (int j = ib; j < i; j++)
        {
            if (Ar[j] > av)
            {
                Ar[j] = av;
            }
        }

        nn = 0; // Последовательность обработана
    }
}

if (nn > 0)
{
    // Если не обработана последняя отрицательная последовательность
    av /= nn;

    for (int j = ib; j < arraySize; j++)
    {
        if (Ar[j] > av)
        {
            Ar[j] = av;
        }
    }
}

// Вывод результатов с выделением цветом
Console.WriteLine("Массив-результат:");
for (int i = 0; i < arraySize; i++)
{
    if (i > 0 && Ar[i] == Ar[i - 1])
    {
        Console.ForegroundColor = ConsoleColor.Green; // Зеленый цвет для пар
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.White;
    }
}

```

```

    }

    Console.WriteLine($"{Ar[i],3} ");
}
Console.WriteLine();

// Подсчет количества пар соседних элементов с одинаковыми значениями
int pairCount = Enumerable.Range(1, arraySize - 1).Count(i => Ar[i] == Ar[i - 1]);
Console.WriteLine($"Количество пар соседних элементов с одинаковыми значениями: {pairCount}");
Console.ForegroundColor = ConsoleColor.White;
}
}

```

```

Введите размер массива: 200
Начальный массив:
-31 47 -48 -43 38 10 8 -38 -33 -49 -39 45 -28 4 11 -50 31 33 23 -50 -38 49 -44 -49 -33 10 16 8 12 -1
34 -24 47 4 37 -17 -48 -43 44 -3 -29 -25 -21 -35 38 40 -49 40 11 -27 -37 17 -49 14 29 42 -24 4 -5 -48
-20 22 2 -31 -10 41 -33 -12 9 -42 -3 1 -48 -18 -41 -34 4 -26 -46 -48 -17 -23 32 -46 47 9 32 -32 9 8
41 28 -50 -49 -39 27 32 33 27 34 -7 -31 46 8 36 1 35 22 -11 -14 5 -2 -44 37 25 -45 2 26 16 49
27 -27 9 -46 16 40 -34 33 2 -2 -3 13 -45 -14 -48 -28 22 -31 45 29 30 6 -26 50 42 -33 -35 -46 5 40
8 -42 -34 -42 5 39 -50 -17 18 -34 -39 -48 34 11 43 22 -12 -15 22 -2 2 -31 19 50 -5 -38 -3 -15 33 0
41 -31 1 -50 -3 -46 -11 47 17 -28 39 1 -42 -33 11 -41 39 39 -50 12

Массив-результат:
-31 47 -48 -45 38 10 8 -39 -39 -49 -39 45 -28 4 11 -50 31 33 23 -50 -44 49 -44 -49 -42 10 16 8 12 -1
34 -24 47 4 37 -36 -48 -43 44 -22 -29 -25 -22 -35 38 40 -49 40 11 -32 -37 17 -49 14 29 42 -24 4 -5 -48
-24 22 2 -31 -20 41 -33 -22 9 -42 -22 1 -48 -35 -41 -35 4 -32 -46 -48 -32 -32 32 -46 47 9 32 -32 9 8
41 28 -50 -49 -46 27 32 33 27 34 -19 -31 46 8 36 1 35 22 -12 -14 5 -23 -44 37 25 -45 2 26 16 49
27 -27 9 -46 16 40 -34 33 2 -2 -3 13 -45 -33 -48 -33 22 -31 45 29 30 6 -26 50 42 -38 -38 -46 5 40
8 -42 -39 -42 5 39 -50 -33 18 -40 -48 34 11 43 22 -13 -15 22 -2 2 -31 19 50 -15 -38 -15 -15 33 0
41 -31 1 -50 -27 -46 -27 47 17 -28 39 1 -42 -37 11 -41 39 39 -50 12

Количество пар соседних элементов с одинаковыми значениями: 6

```

- Завдання 4*
Виконати минуле завдання, використовуючи замість масивів одну з колекцій мови C# (List, LinkedList, ArrayList, чи ін.).

```

using System;
using System.Collections.Generic;
using System.Linq;

class Program
{
    static void Main()
    {
        Console.WriteLine("Введите размер списка: ");
        if (!int.TryParse(Console.ReadLine(), out int listSize) || listSize <= 0)
        {
            Console.WriteLine("Некорректный ввод размера списка.");
            return;
        }

        List<int> list = new List<int>(listSize); // создаем список для обработки

        Random random = new Random(); // создаем генератор случайных чисел

        // Заполнение списка случайными числами
        for (int i = 0; i < listSize; i++)
        {

```

```

    list.Add(random.Next(-50, 51)); // диапазон значений от -50 до 50
}

// Вывод начального списка
Console.WriteLine("Начальный список:");
foreach (int item in list)
{
    Console.Write($"{item,3} ");
}
Console.WriteLine();
Console.WriteLine();

int nn = 0; // количество элементов в последовательности
int ib = 0; // индекс начала последовательности
int av = 0; // среднее значение

for (int i = 0; i < list.Count; i++)
{
    if (list[i] < 0)
    {
        // Обработка отрицательного элемента
        if (nn == 0)
        {
            // Начало последовательности
            ib = i;
            av = list[i];
            nn = 1;
        }
        else
        {
            // Накопление суммы и подсчет количества
            av += list[i];
            nn++;
        }
    }
    else
    {
        // Обработка положительного элемента
        if (nn > 0)
        {
            // Если есть необработанная отрицательная последовательность
            av /= nn; // Усреднение

            // Подсчет пар соседних элементов с одинаковыми значениями
            for (int j = ib; j < i; j++)
            {
                if (list[j] > av)
                {
                    list[j] = av;
                }
            }
        }
    }
}

```

```

        nn = 0; // Последовательность обработана
    }
}
}

if (nn > 0)
{
    // Если не обработана последняя отрицательная последовательность
    av /= nn;

    for (int j = ib; j < list.Count; j++)
    {
        if (list[j] > av)
        {
            list[j] = av;
        }
    }
}

// Вывод результатов с выделением цветом
Console.WriteLine("Список-результат:");
for (int i = 0; i < list.Count; i++)
{
    if (i > 0 && list[i] == list[i - 1])
    {
        Console.ForegroundColor = ConsoleColor.Green; // Зеленый цвет для пар
    }
    else
    {
        Console.ForegroundColor = ConsoleColor.White;
    }

    Console.Write($"{list[i],3} ");
}
Console.WriteLine();

// Подсчет количества пар соседних элементов с одинаковыми значениями
int pairCount = Enumerable.Range(1, list.Count - 1).Count(i => list[i] == list[i - 1]);
Console.WriteLine($"Количество пар соседних элементов с одинаковыми значениями:
{pairCount}");
Console.ForegroundColor = ConsoleColor.White;
}
}

```

```

Введите размер списка: 200
Начальный список:
42 13 -26 9 -48 -17 7 0 -24 -49 -4 25 -12 1 -14 36 32 -21 -39 -21 -27 17 23 0 -29 -5 11 1 46 32
-7 29 -24 47 -12 8 9 48 19 6 -29 -29 47 22 -29 -50 -2 -7 -35 20 17 44 -21 5 -29 -4 40 49 10 9
-36 -11 -17 23 -2 28 -5 16 -13 27 2 -35 -24 -15 17 -21 -3 32 5 -36 -20 11 20 37 -10 -34 -49 49 46 24
-38 -13 -42 25 16 48 -10 -42 -2 -38 -48 -32 -23 23 45 -40 -28 -38 -23 -39 -40 -38 -27 -3 38 -48 11 -9 9 14
42 -32 -36 -13 27 35 0 21 -40 -8 20 -49 29 1 -8 7 -37 41 -13 42 5 -19 27 -21 30 40 25 0 14 23
20 45 -31 0 -13 -34 26 2 27 21 33 20 37 -42 16 5 6 -18 -1 35 0 17 -45 -41 -31 3 35 -19 27 -34
-26 7 -34 11 9 5 -20 -10 8 16 38 -4 13 40 24 24 20 -32 40 14

Список-результат:
42 13 -26 9 -48 -28 7 0 -25 -49 -25 25 -12 1 -14 36 32 -29 -39 -21 -29 17 23 0 -29 -17 11 1 46 32
-7 29 -24 47 -12 8 9 48 19 6 -29 -29 47 22 -29 -50 -24 -35 20 17 44 -21 5 -29 -16 40 49 10 9
-36 -21 -21 23 -2 28 -5 16 -13 27 2 -35 -24 24 17 -21 -12 32 5 -36 -28 11 20 37 -31 -34 -49 49 46 24
-38 -31 -42 25 16 48 -27 -42 -27 -38 -48 -32 -27 23 45 -40 -29 -38 -29 -39 -40 -38 -29 -38 -48 11 -9 9 14
42 -32 -36 -25 27 35 0 21 -40 -28 20 -49 29 1 -8 7 -37 41 -13 42 5 -19 27 -21 30 40 25 0 14 23
20 45 -31 0 -23 -34 26 2 27 21 33 20 37 -42 16 5 6 -18 -9 35 0 17 -45 -41 -39 3 35 -19 27 -34
-30 7 -34 11 9 5 -20 -15 8 16 38 -4 13 40 24 24 20 -32 40 14

Количество пар соседних элементов с одинаковыми значениями: 6

```

Завдання 5

Використовуючи завдання власного варіанта до лабораторної роботи 9 з електронних методичних вказівок з основ програмування та алгоритмічних мов створити проект з реалізації цих завдань засобами мови C#.

Заполнить матрицу случайными числами. Отобразить матрицу симметрично относительно главной диагонали

```
using System;

class Program
{
    static void Main()
    {
        // Создаем и заполняем квадратную матрицу 9x9 от 1 до 81
        int[,] matrix = new int[9, 9];
        int value = 1;
        for (int row = 0; row < 9; row++)
        {
            for (int col = 0; col < 9; col++)
            {
                matrix[row, col] = value;
                value++;
            }
        }

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

        // Создаем и отображаем матрицу, отраженную относительно главной диагонали
        int[,] reflectedMatrix = ReflectMatrix(matrix);
        Console.WriteLine("\nМатрица, отраженная относительно главной диагонали:");
        PrintMatrix(reflectedMatrix);
    }

    // Функция для отображения матрицы
    static void PrintMatrix(int[,] matrix)
    {
        for (int row = 0; row < matrix.GetLength(0); row++)
        {
            for (int col = 0; col < matrix.GetLength(1); col++)
            {
                Console.Write(matrix[row, col] + "\t");
            }
            Console.WriteLine();
        }
    }
}
```

```
// Функция для отражения матрицы относительно главной диагонали
static int[,] ReflectMatrix(int[,] matrix)
{
    int rows = matrix.GetLength(0);
    int cols = matrix.GetLength(1);
    int[,] reflectedMatrix = new int[cols, rows];

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            reflectedMatrix[col, row] = matrix[row, col];
        }
    }

    return reflectedMatrix;
}
}
```

Консоль отладки Microsoft V

Исходная матрица:

1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18
19	20	21	22	23	24	25	26	27
28	29	30	31	32	33	34	35	36
37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54
55	56	57	58	59	60	61	62	63
64	65	66	67	68	69	70	71	72
73	74	75	76	77	78	79	80	81

Матрица, отраженная относительно главной диагонали:

1	10	19	28	37	46	55	64	73
2	11	20	29	38	47	56	65	74
3	12	21	30	39	48	57	66	75
4	13	22	31	40	49	58	67	76
5	14	23	32	41	50	59	68	77
6	15	24	33	42	51	60	69	78
7	16	25	34	43	52	61	70	79
8	17	26	35	44	53	62	71	80
9	18	27	36	45	54	63	72	81

Завдання 6*

Виконати минуле завдання, зробивши матрицю динамічною, задавши її розміри з консолі, використовуючи для цього прямокутний ([,]) двовимірний масив

```
using System;
```

```
class Program
{
    static void Main()
    {
        Console.Write("Введіть кількість строк матриці: ");
        int n = int.Parse(Console.ReadLine());

        Console.Write("Введіть кількість стовбців матриці: ");
        int m = int.Parse(Console.ReadLine());

        // Создаем динамическую матрицу
        int[,] matrix = new int[n, m];
        int value = 1;

        for (int row = 0; row < n; row++)
        {
            for (int col = 0; col < m; col++)
            {
                matrix[row, col] = value;
                value++;
            }
        }

        Console.WriteLine("\nИсходная матрица:");
        PrintMatrix(matrix);

        // Создаем и отображаем матрицу, отраженную относительно главной диагонали
        int[,] reflectedMatrix = ReflectMatrix(matrix);
        Console.WriteLine("\nМатрица, отраженная относительно главной диагонали:");
        PrintMatrix(reflectedMatrix);
    }

    // Функция для отображения матрицы
    static void PrintMatrix(int[,] matrix)
    {
        for (int row = 0; row < matrix.GetLength(0); row++)
        {
            for (int col = 0; col < matrix.GetLength(1); col++)
            {
                Console.Write(matrix[row, col] + "\t");
            }
            Console.WriteLine();
        }
    }
}
```

```

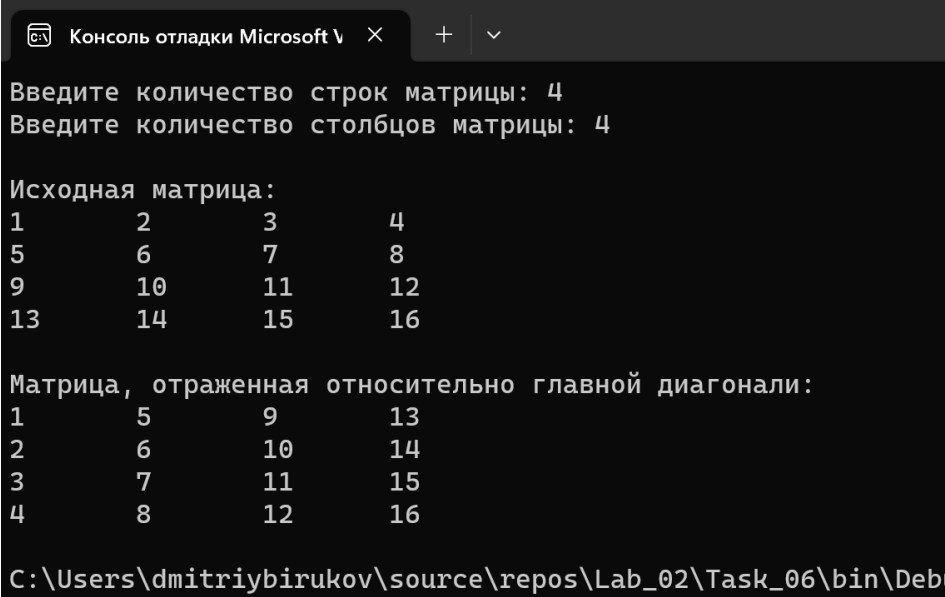
}

// Функция для отражения матрицы относительно главной диагонали
static int[,] ReflectMatrix(int[,] matrix)
{
    int rows = matrix.GetLength(0);
    int cols = matrix.GetLength(1);
    int[,] reflectedMatrix = new int[cols, rows];

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            reflectedMatrix[col, row] = matrix[row, col];
        }
    }

    return reflectedMatrix;
}
}

```



```

Консоль отладки Microsoft V  ×  +  ▾
Введите количество строк матрицы: 4
Введите количество столбцов матрицы: 4

Исходная матрица:
1      2      3      4
5      6      7      8
9      10     11     12
13     14     15     16

Матрица, отраженная относительно главной диагонали:
1      5      9      13
2      6      10     14
3      7      11     15
4      8      12     16

C:\Users\dmitriybirukov\source\repos\Lab_02\Task_06\bin\Deb

```


Завдання 7*

Виконати минуле завдання, зробивши матрицю динамічною, задавши її розміри з консолі, використовуючи для цього зубчастий ([][]) двовимірний масив.

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.Write("Введите количество строк матрицы: ");
```

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

```
        Console.Write("Введите количество столбцов матрицы: ");
```

```
        int m = int.Parse(Console.ReadLine());
```

```
        // Создаем зубчатый двумерный массив
```

```
        int[][] matrix = new int[n][];
```

```
        // Инициализируем каждую строку массива
```

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

```
        {
```

```
            matrix[i] = new int[m];
```

```
        }
```

```
        int value = 1;
```

```
        // Заполняем матрицу числами от 1 до n*m
```

```
        for (int row = 0; row < n; row++)
```

```
        {
```

```
            for (int col = 0; col < m; col++)
```

```
            {
```

```
                matrix[row][col] = value;
```

```
                value++;
```

```
            }
```

```
        }
```

```
        Console.WriteLine("\nИсходная матрица:");
```

```
        PrintMatrix(matrix);
```

```
        // Создаем и отображаем матрицу, отраженную относительно главной  
        диагонали
```

```
        int[][] reflectedMatrix = ReflectMatrix(matrix);
```

```

        Console.WriteLine("\nМатрица, отраженная относительно главной
диагонали:");
        PrintMatrix(reflectedMatrix);
    }

// Функция для отображения зубчатой матрицы
static void PrintMatrix(int[][] matrix)
{
    foreach (int[] row in matrix)
    {
        foreach (int element in row)
        {
            Console.Write(element + "\t");
        }
        Console.WriteLine();
    }
}

// Функция для отражения матрицы относительно главной диагонали
static int[][] ReflectMatrix(int[][] matrix)
{
    int rows = matrix.Length;
    int cols = matrix[0].Length;
    int[][] reflectedMatrix = new int[cols][];

    for (int col = 0; col < cols; col++)
    {
        reflectedMatrix[col] = new int[rows];
        for (int row = 0; row < rows; row++)
        {
            reflectedMatrix[col][row] = matrix[row][col];
        }
    }

    return reflectedMatrix;
}
}

```

```

Введите количество строк матрицы: 5
Введите количество столбцов матрицы: 5

Исходная матрица:
1      2      3      4      5
6      7      8      9      10
11     12     13     14     15
16     17     18     19     20
21     22     23     24     25

Матрица, отраженная относительно главной диагонали:
1      6      11     16     21
2      7      12     17     22
3      8      13     18     23
4      9      14     19     24
5      10     15     20     25

C:\Users\dmitriybirukov\source\repos\Lab_02\Task_07\bin\Deb

```

Завдання 8*

Виконати минуле завдання, використовуючи кольорові можливості консолі для покращення інтерфейсу роботи програми. Наприклад, виділяти різними кольорами знайдені в матриці за завданням елементи чи їхні послідовності, розраховані параметри, тощо.

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.Write("Введіть кількість рядків матриці: ");
```

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

```
        Console.Write("Введіть кількість стовпців матриці: ");
```

```
        int m = int.Parse(Console.ReadLine());
```

```
        // Создаем зубчатый двумерный массив
```

```
        int[][] matrix = new int[n][];
```

```
        // Инициализируем каждую строку массива
```

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

```
        {
```

```
            matrix[i] = new int[m];
```

```
        }
```

```
        int value = 1;
```

```
        // Заполняем матрицу числами от 1 до n*m
```

```
        for (int row = 0; row < n; row++)
```

```
        {
```

```
            for (int col = 0; col < m; col++)
```

```
            {
```

```
                matrix[row][col] = value;
```

```
                value++;
```

```
            }
```

```
        }
```

```
        Console.WriteLine("\nИсходная матрица:");
```

```
        PrintMatrix(matrix);
```

```
        // Создаем и отображаем матрицу, отраженную относительно главной диагонали
```

```
        int[][] reflectedMatrix = ReflectMatrix(matrix);
```

```
Console.WriteLine("\nМатрица, отраженная относительно главной  
диагонали:");
```

```
PrintMatrix(reflectedMatrix);  
}
```

```
// Функция для отображения зубчатой матрицы
```

```
static void PrintMatrix(int[][] matrix)
```

```
{  
    int rows = matrix.Length;  
    int cols = matrix[0].Length;
```

```
    for (int row = 0; row < rows; row++)
```

```
    {  
        for (int col = 0; col < cols; col++)  
        {
```

```
            // Цветной вывод для элементов матрицы
```

```
            Console.ForegroundColor = GetColor(matrix[row][col]);
```

```
            Console.Write(matrix[row][col] + "\t");
```

```
            Console.ResetColor(); // Сброс цвета
```

```
        }  
        Console.WriteLine();
```

```
    }  
}
```

```
// Функция для отражения матрицы относительно главной диагонали
```

```
static int[][] ReflectMatrix(int[][] matrix)
```

```
{  
    int rows = matrix.Length;  
    int cols = matrix[0].Length;  
    int[][] reflectedMatrix = new int[cols][];
```

```
    for (int col = 0; col < cols; col++)
```

```
    {  
        reflectedMatrix[col] = new int[rows];
```

```
        for (int row = 0; row < rows; row++)
```

```
        {  
            reflectedMatrix[col][row] = matrix[row][col];  
        }  
    }
```

```
    return reflectedMatrix;  
}
```

```
// Функция для выбора цвета в зависимости от значения элемента
```

```
static ConsoleColor GetColor(int value)
```

```

{
    // Примеры цветов можно настроить по вашему выбору
    if (value % 2 == 0)
    {
        return ConsoleColor.Blue;
    }
    else
    {
        return ConsoleColor.Red;
    }
}
}

```

```

Введите количество строк матрицы: 7
Введите количество столбцов матрицы: 7

Исходная матрица:
1      2      3      4      5      6      7
8      9      10     11     12     13     14
15     16     17     18     19     20     21
22     23     24     25     26     27     28
29     30     31     32     33     34     35
36     37     38     39     40     41     42
43     44     45     46     47     48     49

Матрица, отраженная относительно главной диагонали:
1      8      15     22     29     36     43
2      9      16     23     30     37     44
3      10     17     24     31     38     45
4      11     18     25     32     39     46
5      12     19     26     33     40     47
6      13     20     27     34     41     48
7      14     21     28     35     42     49

C:\Users\dmitriybirukov\source\repos\Lab_02\Task_08\bin\Debug\

```

Завдання 9

Виконати підсумкове завдання згідно із номером свого варіанту

1. Переставляючи рядки динамічної матриці, розташувати їх у відповідності з ростом суми їх додатних парних елементів.

```
using System;
```

```
class Program
```

```
{
```

```
    static void Main()
```

```
    {
```

```
        Console.Write("Введіть кількість строк матриці: ");
```

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

```
        Console.Write("Введіть кількість стовбців матриці: ");
```

```
        int m = int.Parse(Console.ReadLine());
```

```
        // Создаем и заполняем динамическую матрицу случайными числами
```

```
        int[][] matrix = GenerateRandomMatrix(n, m);
```

```
        Console.WriteLine("\nИсходная матрица:");
```

```
        PrintMatrix(matrix);
```

```
        // Сортируем строки матрицы по сумме положительных четных элементов
```

```
        SortMatrixByPositiveEvenSum(matrix);
```

```
        Console.WriteLine("\nМатрица, отсортированная по сумме положительных  
четных элементов:");
```

```
        PrintMatrix(matrix);
```

```
    }
```

```
    // Генерация случайной матрицы
```

```
    static int[][] GenerateRandomMatrix(int n, int m)
```

```
    {
```

```
        Random random = new Random();
```

```
        int[][] matrix = new int[n][];
```

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

```
        {
```

```
            matrix[i] = new int[m];
```

```
            for (int j = 0; j < m; j++)
```

```

        {
            matrix[i][j] = random.Next(-100, 100); // Задайте диапазон случайных
чисел по вашему выбору
        }
    }

    return matrix;
}

// Функция для отображения зубчатой матрицы
static void PrintMatrix(int[][] matrix)
{
    int rows = matrix.Length;
    int cols = matrix[0].Length;

    for (int row = 0; row < rows; row++)
    {
        for (int col = 0; col < cols; col++)
        {
            Console.Write(matrix[row][col] + "\t");
        }
        Console.WriteLine();
    }
}

// Функция для суммирования положительных четных элементов в строке
static int SumPositiveEvenElements(int[] row)
{
    int sum = 0;

    foreach (int element in row)
    {
        if (element > 0 && element % 2 == 0)
        {
            sum += element;
        }
    }

    return sum;
}

// Функция для сортировки матрицы по сумме положительных четных
элементов
static void SortMatrixByPositiveEvenSum(int[][] matrix)
{

```

```

        Array.Sort(matrix, (row1, row2) =>
SumPositiveEvenElements(row1).CompareTo(SumPositiveEvenElements(row2)));
    }
}

```

```

Введите количество строк матрицы: 5
Введите количество столбцов матрицы: 5

Исходная матрица:
-14      58      74      34      -19
7        -7      22      59      -56
63       -98     46      6       21
-36      24      92      -26     19
-15      84      -4      -44     -43

Матрица, отсортированная по сумме положительных четных элементов:
7        -7      22      59      -56
63       -98     46      6       21
-15      84      -4      -44     -43
-36      24      92      -26     19
-14      58      74      34      -19

C:\Users\dmitriybirukov\source\repos\Lab_02\Task_09\bin\Debug\net6
0.

```


Висновок : у даній лабораторній роботі було виконано 10 завдань різної складності. Були набуті практичні та теоритичні знання з таких тем як :

1.Умовний оператор у мові C#

2.Оператори циклу в мові C#

2.Оператори циклу в мові C# 6.Робота з масивами

Також, розглянуто відображення значень функції $y(x)$ на площині координат. Програма складається з двох основних циклів: зовнішнього та внутрішнього. Зовнішній цикл виконується 5 разів, що відповідає п'яти періодам відображення функції. Внутрішній цикл розраховує значення функції $y(x)$ для різних значень x на кожному з періодів.

Програма виводить координати точок (x, y) на екрані у вигляді таблиці без графічного відображення. Значення функції $y(x)$ розраховуються для трьох різних відрізків залежно від значення x . Величина x змінюється від 0 до 4 з кроком 0.25 на кожному з періодів.

Кожна ітерація внутрішнього циклу виводить координати (x, y) на екран у вигляді таблиці. Після завершення виведення кожного періоду програма очікує натискання клавіші Enter перед продовженням виконання наступного періоду.

Отже, дана програма демонструє спосіб розрахунку та виведення значень функції $y(x)$ для різних значень x на площині координат.