

Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра «МОП ЭВМ»

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «Технологии программирования»

Вариант 6

Студент группы 6ИСб-1

Е.И. Павлюк

Преподаватель

С.Ю. Александров

2018

Содержание

| | |
|--|----|
| Задания | 4 |
| Введение..... | 5 |
| 1 Линейные программы..... | 6 |
| 1.1 Описание программы | 6 |
| 1.2 Текст программы | 6 |
| 1.3 Тестирование программы | 7 |
| 2 Разветвляющиеся вычислительные процессы | 8 |
| 2.1 Описание программы | 8 |
| 2.2 Текст программы | 8 |
| 2.3 Тестирование программы | 10 |
| 3 Организация циклов | 11 |
| 3.1 Описание программы | 11 |
| 3.2 Текст программы | 11 |
| 3.3 Тестирование программы | 12 |
| 4 Простейшие классы | 13 |
| 4.1 Описание программы | 13 |
| 4.2 Текст программы | 13 |
| 4.3 Тестирование программы | 15 |
| 5 Одномерные массивы | 16 |
| 5.1 Описание программы | 16 |
| 5.2 Текст программы | 16 |
| 5.3 Тестирование программы | 18 |
| 6 Двумерные массивы..... | 19 |

| | | |
|------|--|----|
| 6.1 | Описание программы | 19 |
| 6.2 | Текст программы | 19 |
| 6.3 | Тестирование программы | 21 |
| 7 | Строки | 22 |
| 7.1 | Описание программы | 22 |
| 7.2 | Текст программы | 22 |
| 7.3 | Тестирование программы | 23 |
| 8 | Классы и операции..... | 24 |
| 8.1 | Описание программы | 24 |
| 8.2 | Текст программы | 24 |
| 8.3 | Тестирование программы | 26 |
| 9 | Наследование..... | 27 |
| 9.1 | Описание программы | 27 |
| 9.2 | Текст программы | 27 |
| 9.3 | Тестирование программы | 31 |
| 10 | Структуры..... | 36 |
| 10.1 | Описание программы | 36 |
| 10.2 | Текст программы | 36 |
| 10.3 | Тестирование программы | 38 |
| | Заключение | 39 |
| | Список использованных источников | 40 |

Задания

- 1 Линейные программы.
- 2 Разветвляющиеся вычислительные процессы.
- 3 Организация циклов.
- 4 Простейшие классы.
- 5 Одномерные массивы.
- 6 Двумерные массивы.
- 7 Строки.
- 8 Классы и операции.
- 9 Наследование.
- 10 Структуры.

Введение

Язык C# как средство обучения программированию обладает рядом несомненных достоинств. Он хорошо организован, строг, большинство его конструкций логичны и удобны. Развитые средства диагностики и редактирования кода делают процесс программирования приятным и эффективным.

Немаловажно, что C# является не учебным, а профессиональным языком, предназначенным для решения широкого спектра задач, и в первую очередь - в быстро развивающейся области создания распределенных приложений.

1 Линейные программы

Линейной называется программа, все операторы которой выполняются последовательно в том порядке, в котором они записаны.

1.1 Описание программы

Написать программу для расчета по двум формулам

$$z_1 = \cos \alpha + \cos 2\alpha + \cos 6\alpha + \cos 7\alpha; \quad z_2 = 4 \cos \frac{\alpha}{2} \cdot \cos \frac{5}{2} \alpha \cdot \cos 4\alpha.$$

1.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 1.1.

Листинг 1.1 – Текст файла prog1.cs

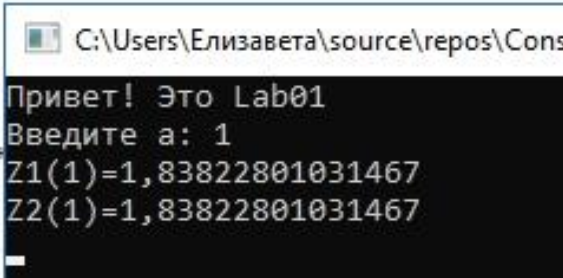
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Lab01
{
    public class Class01_1
    {
        public double Z1(double a)
        {
            return Math.Cos(a) + Math.Cos(2 * a) + Math.Cos(6 * a) + Math.Cos(7 * a);
        }

        public double Z2(double a)
        {
            return 4 * Math.Cos(a / 2) * Math.Cos((5.0 / 2.0) * a) * Math.Cos(4 * a);
        }
    }
    class Program01
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Привет! Это Lab01");
            Console.Write("Введите а: ");
            Class01_1 cl01 = new Class01_1();
            double a = double.Parse(Console.ReadLine());
            Console.WriteLine("Z1({0})={1}\nZ2({0})={2}", a, cl01.Z1(a), cl01.Z2(a));
            Console.ReadLine();
        }
    }
}
```

1.3 Тестирование программы

Результат работы программы приведен на рисунке 1.1.



```
C:\Users\Елизавета\source\repos\Cons
Привет! Это Lab01
Введите a: 1
Z1(1)=1,83822801031467
Z2(1)=1,83822801031467
```

Рисунок 1.1 – Результат работы линейной программы

2 Разветвляющиеся вычислительные процессы

Разветвляющиеся вычислительные процессы – это вычислительные процессы, в которых предусмотрено разветвление выполняемой последовательности действий в зависимости от результата проверки какого-либо условия.

2.1 Описание программы

Написать программу, которая по введенному значению аргумента вычисляет значение функции, заданной в виде графике (рисунок 2.1). Параметр R вводится с клавиатуры.

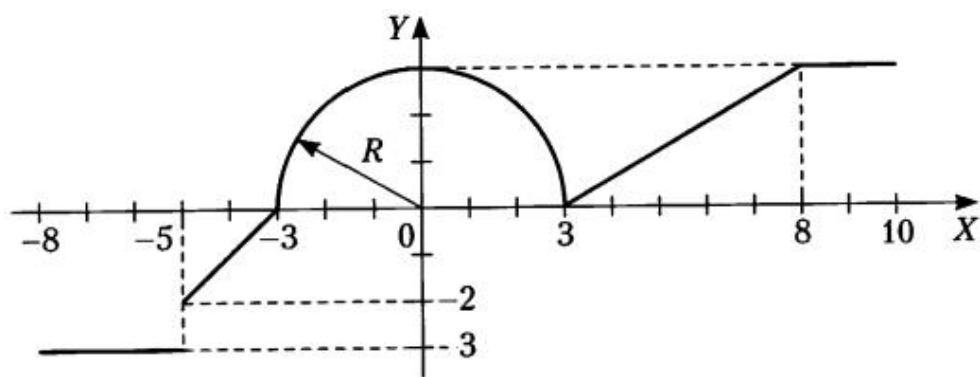


Рисунок 2.1 – График

2.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 2.1.

Листинг 2.1 – Текст файла prog2.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Task
    {
        double y = 0;
```



```

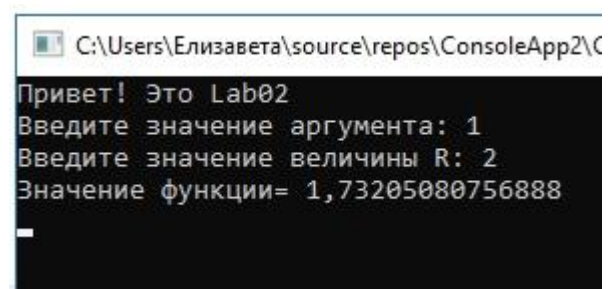
public bool Pop(double x, double r)
{
    if (-8 <= x && x <= -5)
    {
        y = 3;
        return true;
    }
    else if (-5 <= x && x <= -3)
    {
        y = x + 3;
        return true;
    }
    else if (-3 <= x && x <= 3)
    {
        y = Math.Sqrt(Math.Pow(r, 2) - Math.Pow(x, 2));
        return true;
    }
    else if (3 <= x && x <= 8)
    {
        y = -x + 5;
        return true;
    }
    else if (8 <= x && x <= 10)
    {
        y = r;
        return true;
    }
    else
    {
        return false;
    }
}

public double GetDate()
{
    return y;
}
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Привет! Это Lab02");
        while (true)
        {
            Console.Write("Введите значение аргумента: ");
            double x = double.Parse(Console.ReadLine());
            Console.Write("Введите значение величины R: ");
            double r = double.Parse(Console.ReadLine());
            Task t = new Task();
            if (t.Pop(x, r))
            {
                Console.Write("Значение функции= ");
                Console.WriteLine(t.GetDate());
            }
            else
            {
                Console.WriteLine("Неправильное значение аргумента");
                Console.ReadLine();
            }
        }
    }
}

```

2.3 Тестирование программы

Результат работы программы приведен на рисунке 2.2.

A screenshot of a Windows console window. The title bar shows the file path: C:\Users\Елизавета\source\repos\ConsoleApp2\... The console has a black background with white text. The text displayed is: "Привет! Это Lab02", "Введите значение аргумента: 1", "Введите значение величины R: 2", and "Значение функции= 1,73205080756888". A white cursor is visible on the line following the last line of text.

```
C:\Users\Елизавета\source\repos\ConsoleApp2\...  
Привет! Это Lab02  
Введите значение аргумента: 1  
Введите значение величины R: 2  
Значение функции= 1,73205080756888  
_
```

Рисунок 2.2 – Результат работы программы по разветвляющимся
вычислительным процессам

3 Организация циклов

Циклы являются управляющими конструкциями, позволяя в зависимости от определенных условий выполнять некоторое действие множество раз.

3.1 Описание программы

Вычислить и вывести на экран значения функции, заданной с помощью ряда Тейлора, на интервале от $x_{\text{нач}}$ до $x_{\text{кон}}$ с шагом dx с точностью ε .

$$\operatorname{arctg} x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \dots, |x| < 1$$

3.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 3.1.

Листинг 3.1 – Текст файла prog3.cs

```
using System;

namespace Arctan
{
    class Program
    {
        static void Main()
        {
            double eps, x, xlim, dx;
            do
            {
                Console.WriteLine("Введите X0 : |X0| < 1");
            } while (!Double.TryParse(Console.ReadLine(), out x) || Math.Abs(x) >= 1);
            do
            {
                Console.WriteLine("Введите X : X0 < X < 1");
            } while (!Double.TryParse(Console.ReadLine(), out xlim) || xlim <= x);
            do
            {
                Console.WriteLine("Введите dx");
            } while (!Double.TryParse(Console.ReadLine(), out dx) || dx > Math.Abs(1 - x));
            do
            {
                Console.WriteLine("Введите 0 < eps < 1");
            } while (!Double.TryParse(Console.ReadLine(), out eps) || eps >= 1 || eps <= 0);
            for (; x < xlim; x += dx)
            {
                double a = x;
                double s = 0;
                int n = 1;
                for (; Math.Abs(Math.Atan(x) - s) > eps; n++)
```

```

        {
            s = s + a;
            a = -a * x * x * (2.0 * n - 1) / (2 * n + 1);
        }
        if (s == 0) s = x;
        Console.WriteLine("x = {0:f} сумма {1} вычислена с точностью {2} за {3}
итераци" + ((n % 10 == 1 && n / 10 != 1) ? "ю" : (n % 10 > 1 && n % 10 < 5 && n / 10 != 1)
? "и" : "й"), x, s, eps, n);
    }
    Console.ReadKey();
}
}
}

```

3.3 Тестирование программы

Результат работы программы приведен на рисунке 3.1.

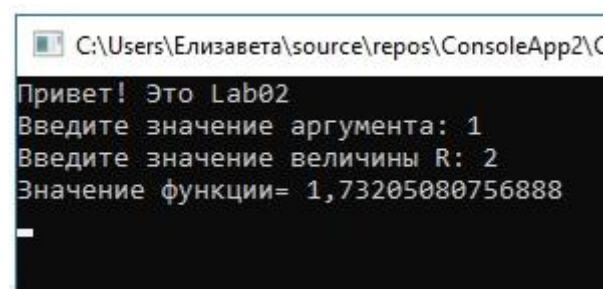


Рисунок 3.1 – Результат работы программы по организации циклов

4 Простейшие классы

Класс является типом данных, определяемым пользователем. Он должен представлять собой одну логическую сущность, например, являться моделью реального объекта или процесса. Элементами класса являются данные и функции, предназначенные для их обработки.

4.1 Описание программы

Составить описание класса для вектора, заданного координатами его концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между двумя векторами. Написать программу, демонстрирующую все разработанные элементы класса.

4.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 4.1.

Листинг 4.1 – Текст файла prog4.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApp1
{
    class Vector
    {
        private double X;
        private double Y;
        private double Z;

        // конструктор без параметров
        private Vector() { }

        //конструктор с параметрами
        public Vector(double x, double y, double z)
        {
            X = x;
            Y = y;
            Z = z;
        }
    }
}
```

```

    }

    //длина вектора
    public double GetLength()
    {
        return Math.Sqrt(X * X + Y * Y + Z * Z);
    }

    //операция сложения
    public static Vector operator +(Vector l, Vector r)
    {
        return new Vector(l.X + r.X, l.Y + r.Y, l.Z + r.Z);
    }

    //вычитание векторов
    public static Vector operator -(Vector l, Vector r)
    {
        return new Vector(l.X - r.X, l.Y - r.Y, l.Z - r.Z);
    }

    //вычисление скалярного произведения двух векторов
    public static double operator *(Vector l, Vector r)
    {
        return (l.X * r.X + l.Y * r.Y + l.Z * r.Z);
    }

    //вычисление косинуса между векторами
    public static double Cos(Vector l, Vector r)
    {
        return (l * r) / (l.GetLength() * r.GetLength());
    }

    public override string ToString()
    {
        return string.Format("{0},{1},{2}", X, Y, Z);
    }
}

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Введите координаты вектора №1:");
        Console.Write("x= ");
        double x1 = double.Parse(Console.ReadLine());
        Console.Write("y= ");
        double y1 = double.Parse(Console.ReadLine());
        Console.Write("z= ");
        double z1 = double.Parse(Console.ReadLine());

        Console.WriteLine("Введите координаты вектора №2:");
        Console.Write("x= ");
        double x2 = double.Parse(Console.ReadLine());
        Console.Write("y= ");
        double y2 = double.Parse(Console.ReadLine());
        Console.Write("z= ");
        double z2 = double.Parse(Console.ReadLine());

        Vector v1 = new Vector(x1, y1, z1);
        Vector v2 = new Vector(x2, y2, z2);
        Vector v3 = v1 + v2;
        Vector v4 = v1 - v2;
        double m = v2 * v3;
        double c = Vector.Cos(v1, v2);
    }
}

```

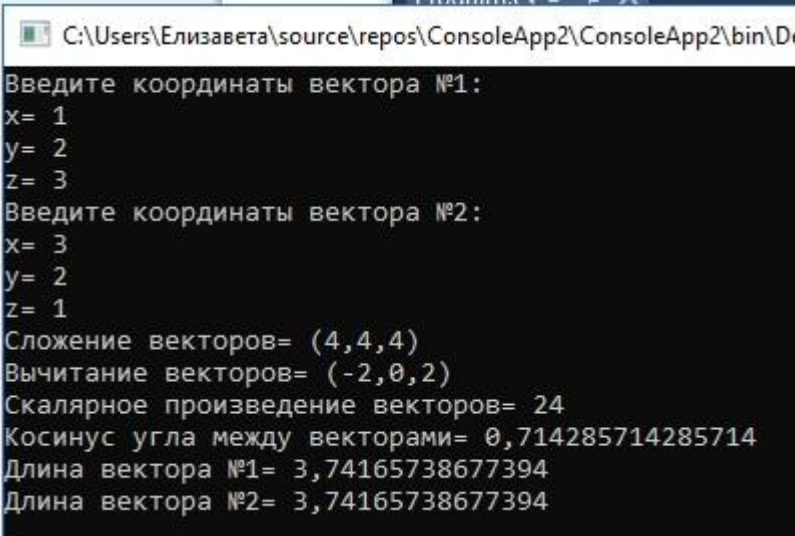
```

        Console.WriteLine("Сложение векторов= " + v3);
        Console.WriteLine("Вычитание векторов= " + v4);
        Console.WriteLine("Скалярное произведение векторов= " + m);
        Console.WriteLine("Косинус угла между векторами= " + c);
        Console.WriteLine("Длина вектора №1= " + v1.GetLength());
        Console.WriteLine("Длина вектора №2= " + v2.GetLength());
        Console.ReadKey();
    }
}

```

4.3 Тестирование программы

Результат работы программы приведен на рисунке 4.1.



The screenshot shows a console window titled "C:\Users\Елизавета\source\repos\ConsoleApp2\ConsoleApp2\bin\De". The output text is as follows:

```

Введите координаты вектора №1:
x= 1
y= 2
z= 3
Введите координаты вектора №2:
x= 3
y= 2
z= 1
Сложение векторов= (4,4,4)
Вычитание векторов= (-2,0,2)
Скалярное произведение векторов= 24
Косинус угла между векторами= 0,714285714285714
Длина вектора №1= 3,74165738677394
Длина вектора №2= 3,74165738677394

```

Рисунок 4.1 – Результат работы программы по простейшим классам

5 Одномерные массивы

До настоящего момента использовали в программах простые переменные. При этом каждой области памяти, выделенной для хранения одной величины, соответствует своё имя. Если переменных много, программа, предназначенная для их обработки, получается длинной и однообразной. Поэтому в любом процедурном языке есть понятие массива – ограниченной совокупности однотипных величин.

Элементы массива имеют одно и то же имя, а различаются порядковым номером (индексом). Это позволяет компактно записывать множество операций с помощью циклов.

5.1 Описание программы

В одномерном массиве, состоящем из n вещественных элементов, вычислить:

- *минимальный элемент массива;*
- *сумму элементов массива, расположенных между первым и последним положительными элементами.*

Преобразовать массив таким образом, чтобы сначала располагались все элементы, равные нулю, а потом – все остальные.

5.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 5.1.

Листинг 5.1 – Текст файла prog5.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApp1
{
    class Mas
    {
```



```

private int[] array;

public Mas(int sizeArr)
{
    array = new int[sizeArr];
    for (int i = 0; i < array.Length; i++)
    {
        Console.Write(string.Format("Введите {0}й элемент массива: ", i + 1));
        array[i] = int.Parse(Console.ReadLine());
    }
}

public void showArray()
{
    for (int i = 0; i < array.Length; i++)
    {
        Console.Write(string.Format(array[i] + " "));
    }
    Console.WriteLine("\n");
}

public void sortArr()
{
    for (int i = 0; i < array.Length; i++)
    {
        if (array[i] == 0)
        {
            for (int j = i; j > 0; j--)
            {
                array[j] = array[j - 1];
            }
            array[0] = 0;
        }
    }
}

public int minEl()
{
    int min_elem = array.Min();
    return min_elem;
}

public int sumBetweenFirstSecond()
{
    int first_index = 0;
    int last_index = array.Length - 1;

    //нахождение индексов элементов
    for (int i = 0; i < array.Length; i++)
    {
        if ((array[i] < 0) && (first_index == 0)) first_index++;
        if ((array[array.Length - i - 1] < 0) && (last_index == array.Length))
last_index--;
    }
    int sum = 0;
    for (int i = first_index; i < last_index; i++)
        sum += array[i];
    return sum;
}

}

class Program
{
    static void Main(string[] args)
    {

```

```

        Console.Write("Введите размер массива: ");
        int sizeArr = int.Parse(Console.ReadLine());

        Mas Mass = new Mas(sizeArr);
        Mass.showArray();

        Console.WriteLine("Min={0}", Mass.minEl());

        Console.WriteLine("Сумма между первым положительным и последним положительным
элементом ={0}", Mass.sumBetweenFirstSecond());

        Mass.sortArr();
        Mass.showArray();

        Console.ReadLine();
    }
}

```

5.3 Тестирование программы

Результат работы программы приведен на рисунке 5.1.

```

C:\Users\Елизавета\source\repos\ConsoleApp2\ConsoleApp2\bin\Debug\ConsoleApp2.exe
Введите размер массива: 4
Введите 1й элемент массива: 1
Введите 2й элемент массива: 0
Введите 3й элемент массива: -2
Введите 4й элемент массива: 2
1 0 -2 2
Min=-2
Сумма между первым положительным и последним положительным элементом =-2
0 1 -2 2

```

Рисунок 5.1 – Результат работы программы по одномерным массивам

6 Двумерные массивы

Двумерный массив - это одномерный массив, элементами которого являются одномерные массивы. Другими словами, это набор однотипных данных, имеющий общее имя, доступ к элементам которого осуществляется по двум индексам

6.1 Описание программы

Дана целочисленная прямоугольная матрица. Определить:

- *сумму элементов в тех столбцах, которые не содержат отрицательных элементов;*
- *номера строк и столбцов всех седловых точек матрицы.*

6.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 6.1.

Листинг 6.1 – Текст файла prog6.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace ConsoleApp2
{
    public class Mas2
    {
        public static void matrix(int[,] s, int stlb, int strk) //матрица
        {
            Random rand = new Random();
            for (int j = 0; j < strk; j++)
            {
                for (int i = 0; i < stlb; i++)
                {
                    s[j, i] = rand.Next(-10, 10);
                }
            }
        }
        public static void ishod(int[,] s, int stlb, int strk)
        {
            for (int j = 0; j < strk; j++)
            {
                for (int i = 0; i < stlb; i++)
                {
                    Console.Write(String.Format(" {0}", s[j, i]));
                }
            }
        }
    }
}
```

```

    }
    Console.WriteLine("\n");
}

}

public static void minuselement(int[,] s, int stlb, int strk) //поиск элементов
меньше 0 и вывод их на экран
{
    Console.WriteLine("Сумма элементов в строках:");
    for (int j = 0; j < strk; j++)
    {
        int k = 0;
        for (int i = 0; i < stlb; i++)
        {
            if (s[j, i] < 0)
            {
                for (int q = 0; q < stlb; q++)
                {
                    k = k + s[j, q];
                }
                Console.WriteLine(String.Format("Номер строки: {0} Сумма элементов:
{1}", j + 1, k));
                break;
            }
        }
    }
}

public static void sedltoch(int[,] s, int stlb, int strk)
{
    int k = 0;
    Console.WriteLine("Седловые точки:");
    for (int j = 0; j < strk; j++)
    {
        int imin = 0, jmin = 0, min = 11, imax = 0, jmax = 0, max = -11;

        for (int i = 0; i < stlb; i++)
        {
            if (s[j, i] < min)
            {
                min = s[j, i]; imin = i; jmin = j;
            }
        }
        //минимальный элемент в строке
        for (int o = 0; o < strk; o++)
        {
            if (s[o, imin] > max)
            {
                max = s[o, imin]; imax = imin; jmax = o;
            }
        }
        //максимальный элемент в данном столбце
        if (s[jmin, imin] == s[jmax, imax])
        {
            k = 1; Console.WriteLine(String.Format("A{0}{1} - седловая точка", jmin
+ 1, imin + 1));
        }
    }
    //сравнение этих элементов
    if (k == 0)
    {
        Console.WriteLine("Таких точек нет");
    }
}

```

```

    }
}
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Введите кол-во столбцов");
        int y = int.Parse(Console.ReadLine());
        Console.WriteLine("Введите кол-во строк");
        int z = int.Parse(Console.ReadLine());
        int[,] mas = new int[z, y];
        Mas2.matrix(mas, z, y); //матрица
        Console.WriteLine("\n");
        Mas2.ishod(mas, z, y); //вывод исходной матрицы
        Console.WriteLine("\n");
        Mas2.minuselement(mas, z, y); //поиск элементов меньше 0 и вывод их на экран
        Console.WriteLine("\n");
        Mas2.sedltoch(mas, z, y); //поиск седловых точек
        Console.ReadLine();
    }
}
}

```

6.3 Тестирование программы

Результат работы программы приведен на рисунке 6.1.

```

C:\Users\Елизавета\source\repos\ConsoleApp2\ConsoleApp2\
Введите кол-во столбцов
3
Введите кол-во строк
3

1 -3 0
9 -6 0
2 1 1

Сумма элементов в строках:
Номер строки: 1 Сумма элементов: -2
Номер строки: 2 Сумма элементов: 3

Седловые точки:
A32 - седловая точка

```

Рисунок 6.1 – Результат работы программы по двумерным массивам

7 Строки

Тип `string`, предназначенный для работы со строками символов в кодировке Unicode, является встроенным типом C#. Ему соответствует базовый класс `System.String` библиотеки .NET.

Несмотря на то что строки являются ссылочным типом данных, на равенство и неравенство проверяются не ссылки, а значения строк. Строки равны, если имеют одинаковое количество символов и совпадают посимвольно.

7.1 Описание программы

Написать программу, которая считывает текст из файла и выводит на экран только предложения, не содержащие запятых.

7.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 7.1.

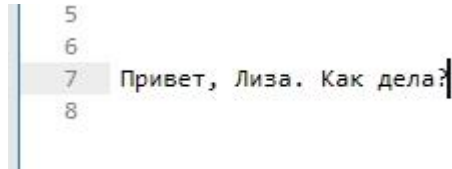
Листинг 7.1 – Текст файла `prog7.cs`

```
using System;
using System.IO;
using System.Linq;

namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine(" Введите название файла");
            string fileName = Console.ReadLine();
            FileStream stream = new FileStream(fileName, FileMode.Open);
            StreamReader reader = new StreamReader(stream);
            string text = reader.ReadToEnd();
            string[] sentences = text.Split(new[] { '.', '!', '?' });
            foreach (string sentence in sentences)
            {
                if (!sentence.Contains(','))
                    Console.WriteLine(sentence.Trim());
            }
            Console.ReadLine();
        }
    }
}
```

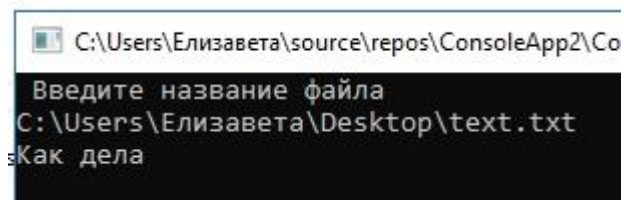
7.3 Тестирование программы

Результат работы программы при считывании текста из файла *.txt (рисунок 7.1) приведён на рисунке 7.2.



```
5  
6  
7 Привет, Лиза. Как дела:  
8
```

Рисунок 7.1 – Текст из файла *.txt



```
C:\Users\Елизавета\source\repos\ConsoleApp2\Co  
Введите название файла  
C:\Users\Елизавета\Desktop\text.txt  
Как дела
```

Рисунок 7.2 - Результат работы программы по строкам

8 Классы и операции

C# позволяет переопределить действие большинства операций так, чтобы при использовании с объектами конкретного класса они выполняли заданные функции. Это даёт возможность применять экземпляры собственных типов данных в составе выражений таким же образом, как стандартных. Определение собственных операций класса часто называют перегрузкой операций.

В C# существуют три вида операций класса: унарные, бинарные и операции преобразования типа.

8.1 Описание программы

Описать класс «домашняя библиотека». Предусмотреть возможность работы с произвольным числом книг, поиска книги по какому-либо признаку (по автору, по году издания или категории), добавления книг в библиотеку, удаления книг из неё, доступа к книге по номеру.

Написать программу, демонстрирующую все разработанные элементы класса.

8.2 Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 8.1.

Листинг 8.1 – Текст файла prog8.cs

```
using System;
using System.Collections.Generic;
namespace ConsoleApp1
{
    public class HomeLibrary
    {
        public int Number { get; set; }
        public string Author { get; set; }
        public int Year { get; set; }
        public string Category { get; set; }
        List<HomeLibrary> homeLibraryList = new List<HomeLibrary>();
        public void Add(HomeLibrary homeLibrary)
        {
            homeLibraryList.Add(homeLibrary);
        }
        public void Remove(int pointer)
```



```

    {
        homeLibraryList.RemoveAt(pointer);
    }
    public override string ToString()
    {
        return Number + " " + Author + " " + Year + " " + Category;
    }
    public void ShowScreen()
    {
        foreach (var VARIABLE in homeLibraryList)
        {
            Console.WriteLine(VARIABLE.ToString());
        }
        Console.WriteLine("\n");
    }
    public void Search(string str, int number)
    {
        foreach (var VARIABLE in homeLibraryList)
        {
            if (number == 1 && VARIABLE.Author == str)
            {
                Console.WriteLine(VARIABLE.ToString());
            }
            else if (number == 2 && VARIABLE.Year == Convert.ToInt32(str))
            {
                Console.WriteLine(VARIABLE.ToString());
            }
            else if (number == 3 && VARIABLE.Category == str)
            {
                Console.WriteLine(VARIABLE.ToString());
            }
            else if (number == 4 && VARIABLE.Number == Convert.ToInt32(str))
            {
                Console.WriteLine(VARIABLE.ToString());
            }
        }
    }
}
class Program
{
    static void Main(string[] args)
    {
        HomeLibrary homeLibrary = new HomeLibrary();
        homeLibrary.Add(new HomeLibrary { Number = 1, Author = "Ессенин", Year = 1900,
Category = "Стихи" });
        homeLibrary.Add(new HomeLibrary { Number = 2, Author = "Пушкин", Year = 1910,
Category = "Рассказы" });
        homeLibrary.Add(new HomeLibrary { Number = 3, Author = "Лермонтов", Year = 1920,
Category = "Басни" });
        homeLibrary.Add(new HomeLibrary { Number = 4, Author = "Достоевский", Year =
1910, Category = "Стихи" });
        homeLibrary.ShowScreen();
        Console.WriteLine("Удаляем данные о 3-ем авторе...\n");
        homeLibrary.Remove(3 - 1);
        homeLibrary.ShowScreen();
        Console.WriteLine("По какому критерию будем производить поиск? (По автору-1, год
издания-2, по категории-3, " + "по номеру -4)");
        int number = int.Parse(Console.ReadLine());
        switch (number)
        {
            case 1:
                Console.WriteLine("Введите писателя");
                homeLibrary.Search(Console.ReadLine(), number);
                break;

```

```

        case 2:
            Console.WriteLine("Введите год издания");
            homeLibrary.Search(Console.ReadLine(), number);
            break;
        case 3:
            Console.WriteLine("Введите категорию");
            homeLibrary.Search(Console.ReadLine(), number);
            break;
        case 4:
            Console.WriteLine("Введите номер книги");
            homeLibrary.Search(Console.ReadLine(), number);
            break;
        default:
            Console.WriteLine("Не правильный критерий");
            break;
    }
    Console.ReadLine();
}
}
}

```

8.3 Тестирование программы

Результат работы программы приведен на рисунке 8.1.

```

C:\Users\mistel\Source\ConsoleApp1\bin\Debug\ConsoleApp1.exe
1 Есенин 1900 Стихи
2 Пушкин 1910 Рассказы
3 Лермонтов 1920 Басни
4 Достоевский 1910 Стихи

Удаляем данные о 3-ем авторе...

1 Есенин 1900 Стихи
2 Пушкин 1910 Рассказы
4 Достоевский 1910 Стихи

По какому критерию будем производить поиск? (По автору-1, год издания-2, по категории-3, по номеру -4)
2
Введите год издания
1910
2 Пушкин 1910 Рассказы
4 Достоевский 1910 Стихи

```

Рисунок 8.1 - Результат работы программы по классам и операциям

9 Наследование

Управлять большим количеством разрозненных классов довольно сложно. С этой проблемой можно справиться путём упорядочивания и ранжирования классов, то есть объединяя общие для нескольких классов свойства в одном классе и используя его в качестве базового.

Эту возможность предоставляет механизм наследования, который является мощнейшим инструментом ООП. Он позволяет строить иерархии, в которых классы-потомки получают свойства классов-предков и могут дополнять их или заменять.

9.1 Описание программы

Создать класс Point (точка). На его основе создать классы ColoredPoint и Line (линия). На основе класса Line создать классы ColoredLine и PolyLine (многоугольник). В классах описать следующие элементы:

- *конструкторы с параметрами и конструкторы по умолчанию;*
- *свойства для установки и получения значений всех координат, а также для изменения цвета и получения текущего цвета;*
- *для линий – методы изменения угла поворота линий относительно первой точки;*
- *для многоугольника – метод масштабирования.*

9.2 Текст программы

Проект состоит из 3 файлов исходного кода, которые приведены в листингах 9.1, 9.2, 9.3.

Листинг 9.1 – Текст файла prog9.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Lab9L
```

```

{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}

```

Листинг 9.2 – Текст файла Form1.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace Lab9L
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            gr = this.CreateGraphics();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            gr.Clear(Color.FromArgb(240, 240, 240));
            Point p1 = new Point(3, 3);
            p1.Draw(600, 500);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            gr.Clear(Color.FromArgb(240, 240, 240));
            Line line = new Line(600, 500, 700, 500);
            line.Draw();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            gr.Clear(Color.FromArgb(240, 240, 240));
            ColoredPoint colPoint = new ColoredPoint(3, 3, 58, 226, 206);
            colPoint.Draw(600, 500);
        }

        private void button9_Click(object sender, EventArgs e)
        {
            gr.Clear(Color.FromArgb(240, 240, 240));
            ColoredLine colline = new ColoredLine(600, 500, 700, 500, 58, 226, 206);
            colline.Draw();
        }

        private void button3_Click(object sender, EventArgs e)
        {

```

```

        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon(200,150);
        plg.Draw(400, 500);

    }

    private void button4_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon();
        plg.Width -= 8;
        plg.Heigh -= 5;
        plg.Draw(400, 500);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon();
        plg.Width += 8;
        plg.Heigh += 5;
        plg.Draw(400, 500);
    }

    private void button6_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Line line = new Line();

        if (Line._StartX == 600)
        {
            line.StartX = 650;
            line.StartY = 450;

            line.EndX = 650;
            line.EndY = 550;

            line.Draw();
        }
        else
        {
            line.StartX = 600;
            line.StartY = 500;

            line.EndX = 700;
            line.EndY = 500;

            line.Draw();
        }
    }
}

```

Листинг 9.3 – Текст файла Form1.Designer.cs

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace Lab9L
{
    public partial class Form1 : Form

```

```

{
    public Form1()
    {
        InitializeComponent();
        gr = this.CreateGraphics();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Point p1 = new Point(3, 3);
        p1.Draw(600, 500);
    }

    private void button2_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Line Line = new Line(600, 500, 700, 500);
        Line.Draw();
    }

    private void button8_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        ColoredPoint colPoint = new ColoredPoint(3, 3, 58, 226, 206);
        colPoint.Draw(600, 500);
    }

    private void button9_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        ColoredLine colline = new ColoredLine(600, 500, 700, 500, 58, 226, 206);
        colline.Draw();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon(200,150);
        plg.Draw(400, 500);
    }

    private void button4_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon();
        plg.Width -= 8;
        plg.Heigh -= 5;
        plg.Draw(400, 500);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
        Polygon plg = new Polygon();
        plg.Width += 8;
        plg.Heigh += 5;
        plg.Draw(400, 500);
    }

    private void button6_Click(object sender, EventArgs e)
    {
        gr.Clear(Color.FromArgb(240, 240, 240));
    }
}

```

```

        Line line = new Line();

        if (Line._StartX == 600)
        {
            line.StartX = 650;
            line.StartY = 450;

            line.EndX = 650;
            line.EndY = 550;

            line.Draw();
        }
        else
        {
            line.StartX = 600;
            line.StartY = 500;

            line.EndX = 700;
            line.EndY = 500;

            line.Draw();
        }
    }
}

```

9.3 Тестирование программы

Результат работы программы приведен на рисунках 9.1 - 9.8.

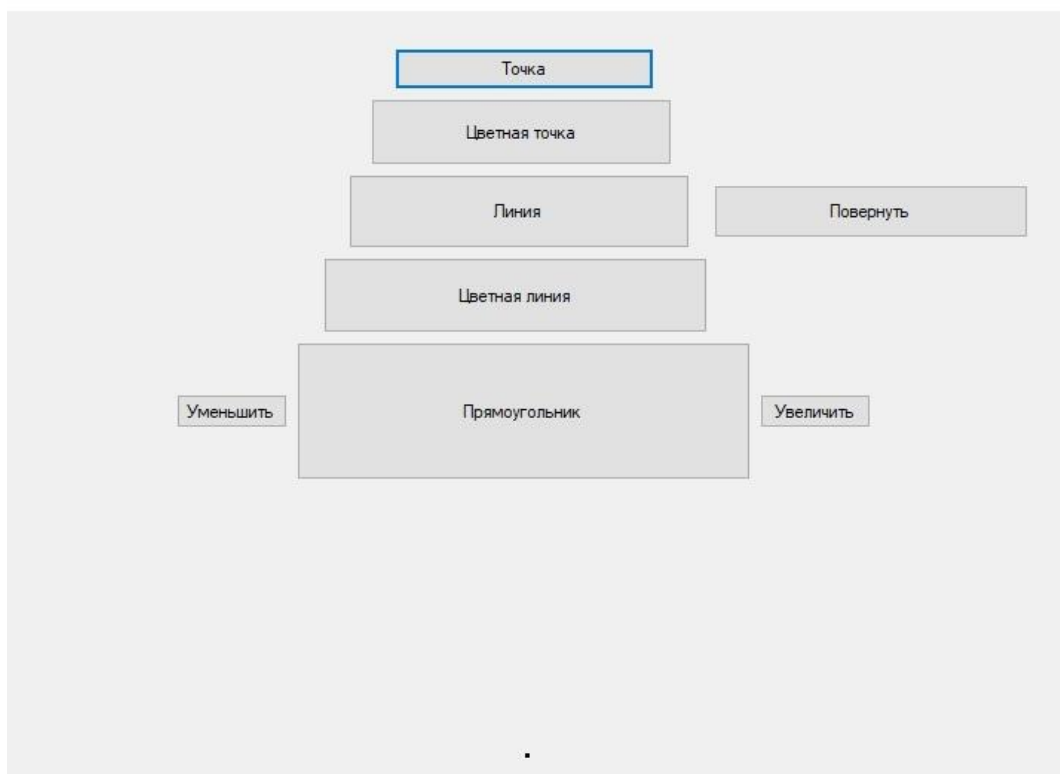


Рисунок 9.1 - Результат работы программы по классу «точка»

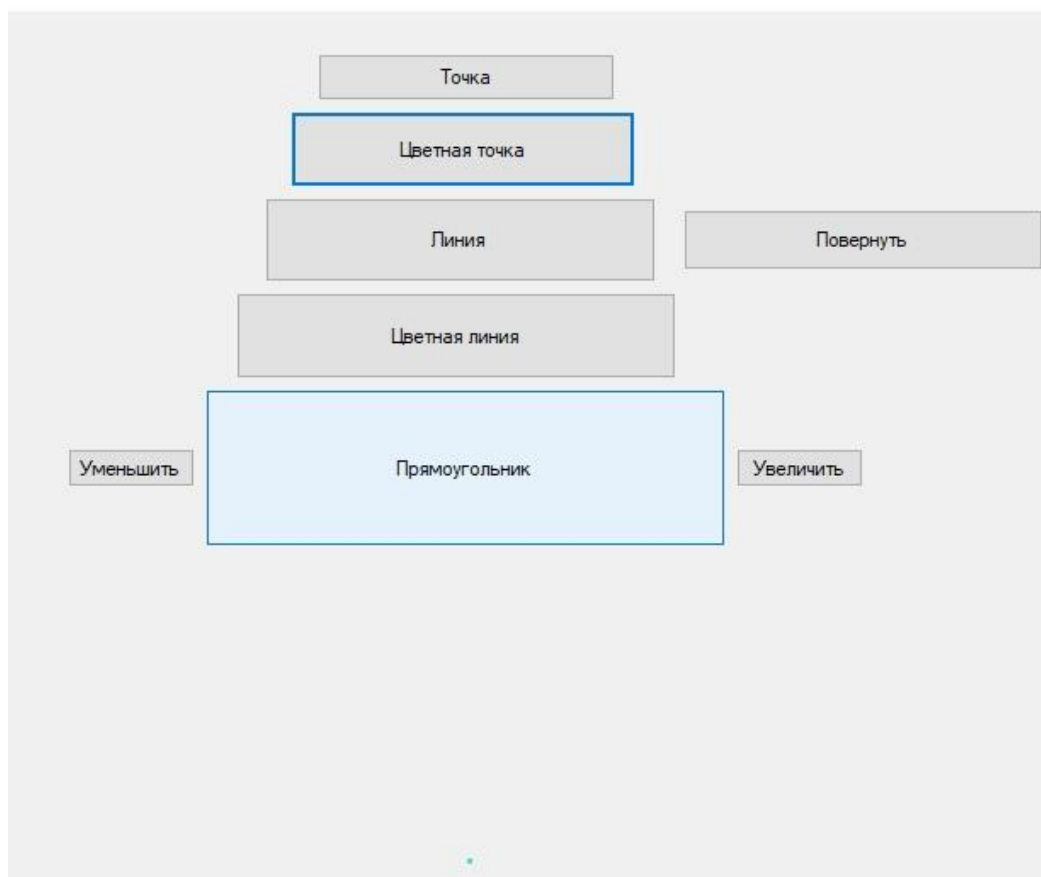


Рисунок 9.2 - Результат работы программы по классу «цветная точка»

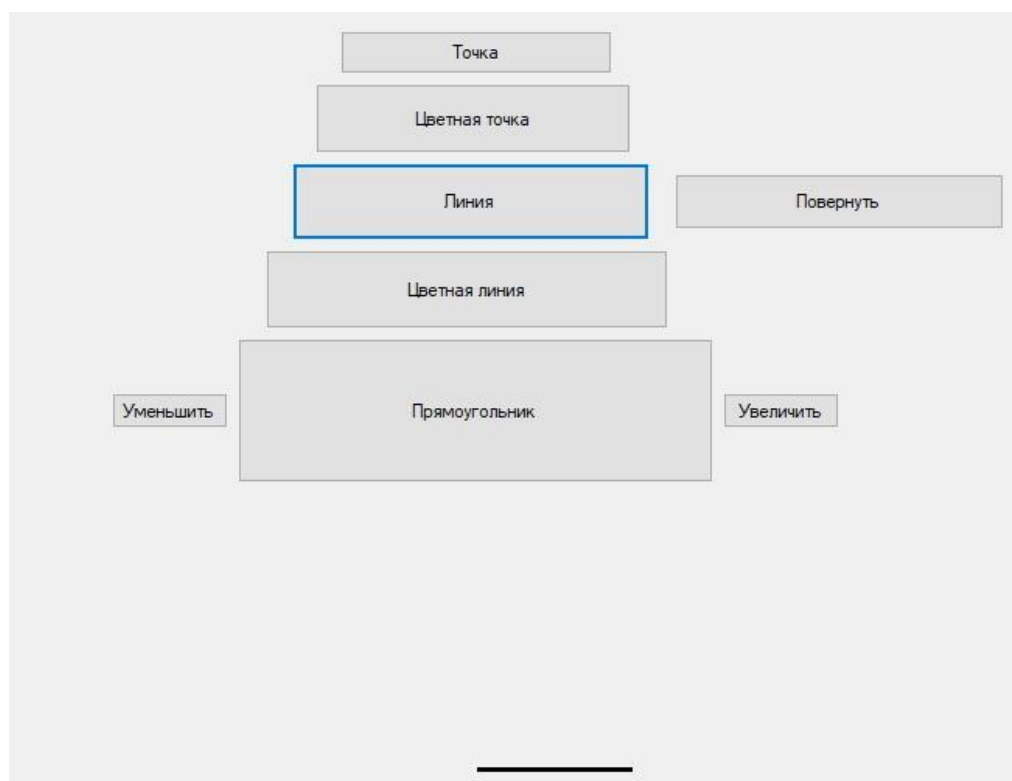


Рисунок 9.3 - Результат работы программы по классу «линия»

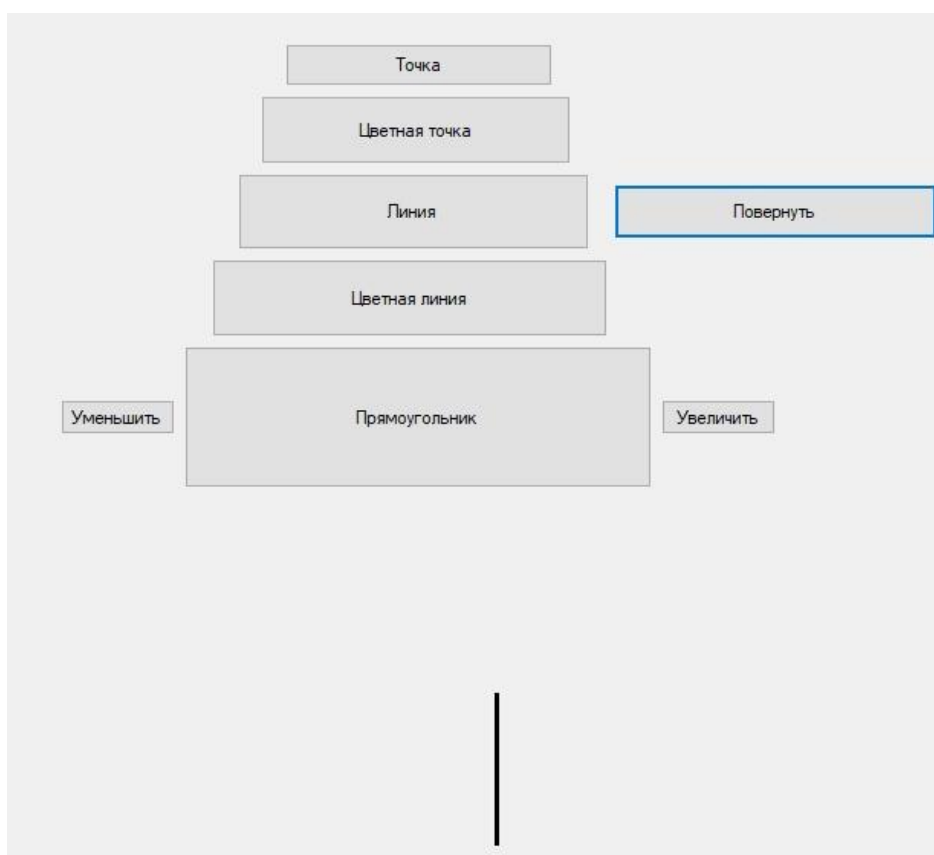


Рисунок 9.4 - Результат работы программы по методу «повернуть линию»

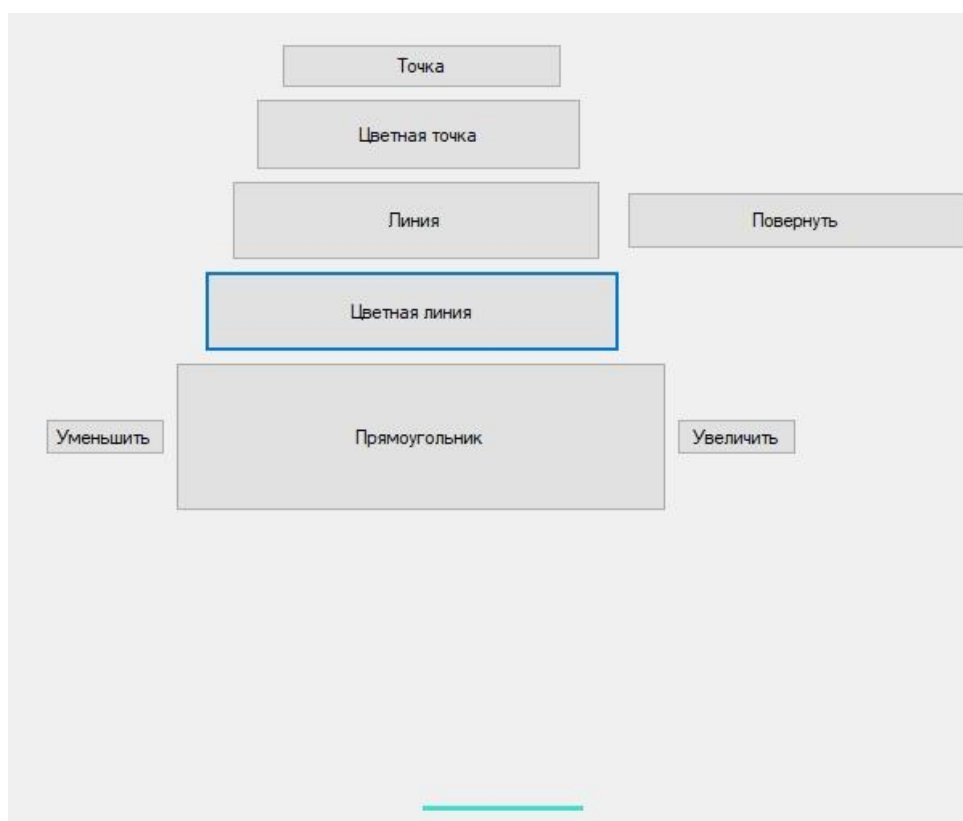


Рисунок 9.5 - Результат работы программы по классу «цветная линия»

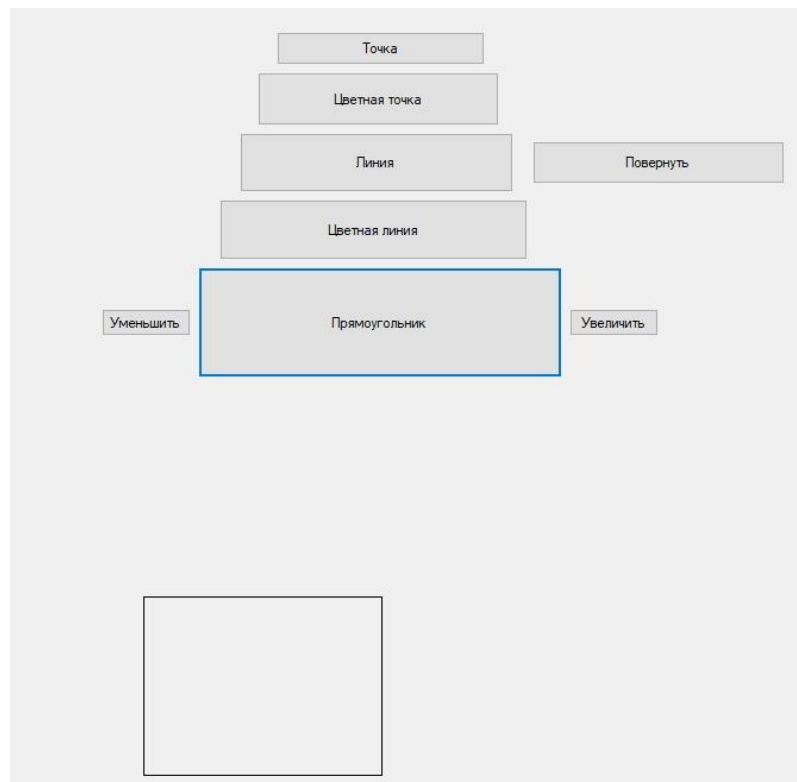


Рисунок 9.6 - Результат работы программы по классу «прямоугольник»

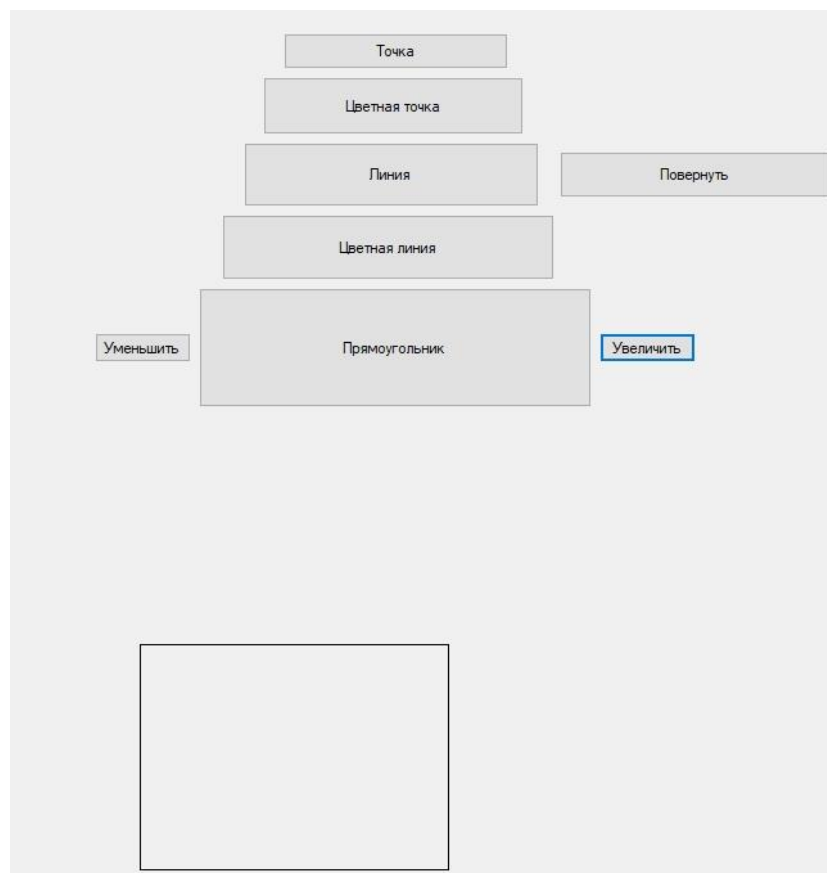


Рисунок 9.7 - Результат работы программы по масштабированию
прямоугольника (увеличить)

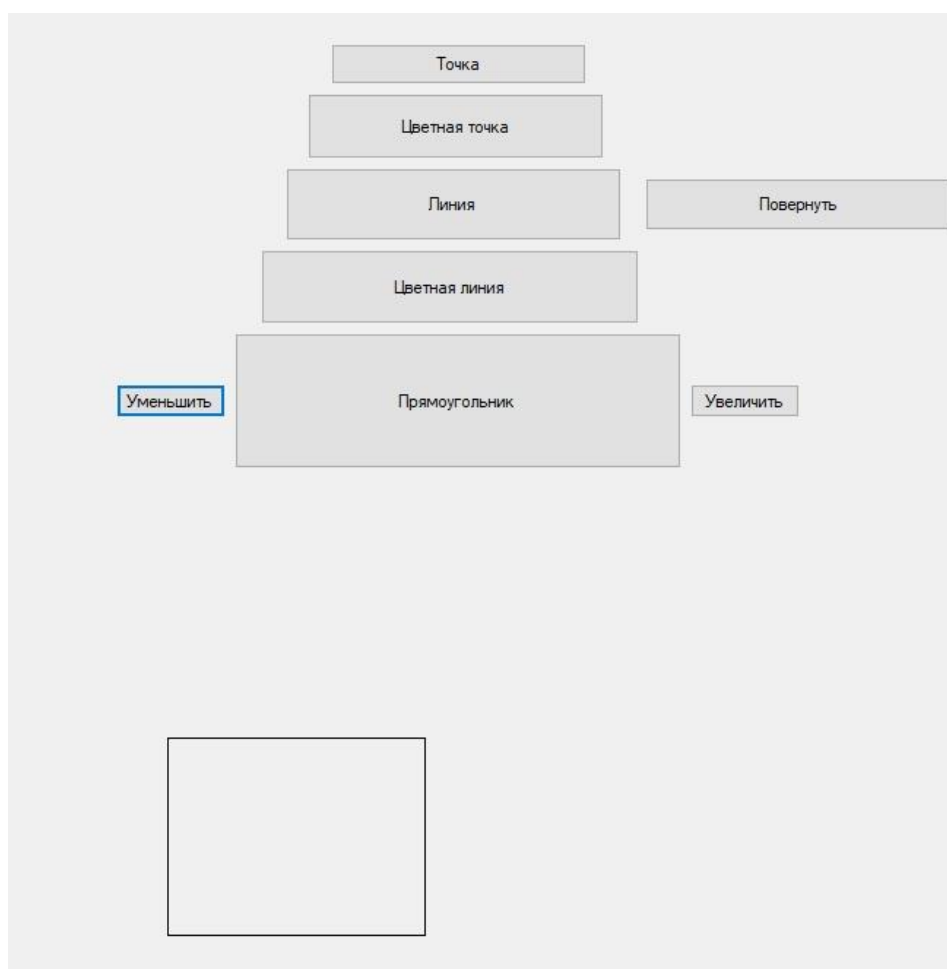


Рисунок 9.8 - Результат работы программы по масштабированию прямоугольника (уменьшить)

10 Структуры

Структура – тип данных, аналогичный классу, но имеющий ряд важных отличий от него:

- структура является значимым, а не ссылочным типом данных, то есть экземпляр структуры хранит значения своих элементов, а не ссылки на них, и располагается в стеке, а не в хипе;
- структура не может участвовать в иерархиях наследования, она может только реализовывать интерфейсы;
- в структуре запрещено определять конструктор по умолчанию, поскольку он определен неявно и присваивает всем её элементам значения по умолчанию;
- в структуре запрещено определять деструкторы, поскольку это бессмысленно.

10.1 Описание программы

Описать структуру WORKER, содержащую следующие поля:

- *фамилия и инициалы работника;*
- *название занимаемой должности;*
- *год поступления на работу.*

Написать программу, выполняющую следующие действия:

- *ввод с клавиатуры данных в массив, состоящий из двух структур типа WORKER (записи должны быть упорядочены по алфавиту);*
- *вывод на экран фамилий работников, стаж работы которых превышает значение, введенное с клавиатуры (если таких работников нет, вывести соответствующее сообщение).*

Текст программы

Проект состоит из одного файла исходного кода, который приведен в листинге 10.1.

Листинг 10.1 – Текст файла prog10.cs

```
using System;

namespace Lab10
{
    class Program
    {
        const int NUMBER_OF_WORKER = 2;

        public static void Main(string[] args)
        {
            Worker[] worker = new Program.Worker[NUMBER_OF_WORKER];

            GetValues(worker);
            Array.Sort(worker);

            Console.WriteLine("Все работники");
            for (int i = 0; i < worker.Length; i++)
                Console.WriteLine(worker[i].Name + " " + worker[i].YearArrival + " " +
worker[i].Post);
            Console.WriteLine();

            int matches = 0;
            Console.WriteLine("Поиск работников");
            Console.Write("Введите стаж: ");
            int DestinationToCompare = Convert.ToInt32(Console.ReadLine());

            for (int i = 0; i < worker.Length; i++)
            {
                if (2018 - worker[i].YearArrival > DestinationToCompare)
                    Console.WriteLine(worker[i].Name + " " + worker[i].YearArrival + " " +
worker[i].Post);
                matches += 1;
            }

            if (matches == 0)
                Console.WriteLine("Работники не найдены...");

            Console.ReadLine();
        }

        public struct Worker : IComparable
        {
            public string Name;
            public string Post;
            public int YearArrival;

            public int CompareTo(object obj)
            {
                Worker arr = (Worker)obj;
                return this.Name.CompareTo(arr.Name);
            }
        }

        static void GetValues(Worker[] Workers)
        {
            for (int i = 0; i < NUMBER_OF_WORKER; i++)
            {
                Worker worker = new Worker();
            }
        }
    }
}
```

```

        Console.WriteLine("Ввод данных о " + (i + 1) + " сотруднике");
        Console.Write("Введите имя: ");
        worker.Name = Console.ReadLine();
        Console.Write("Должность: ");
        worker.Post = Console.ReadLine();
        Console.Write("Год поступления на работу: ");
        worker.YearArrival = Convert.ToInt32(Console.ReadLine());

        Workers[i] = worker;
        Console.WriteLine();
    }
}
}
}
}

```

10.3 Тестирование программы

Результат работы программы приведен на рисунке 10.1.

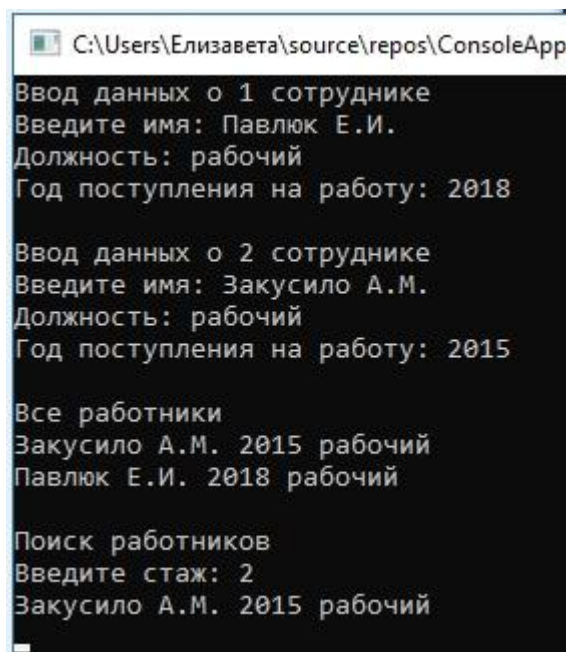


Рисунок 10.1 - Результат работы программы по структурам

Заключение

В ходе изучения дисциплины «Технологии программирования» по изучению языка программирования С# были рассмотрены такие темы как:

- 1 Линейные программы.
- 2 Разветвляющиеся вычислительные процессы.
- 3 Организация циклов.
- 4 Простейшие классы.
- 5 Одномерные массивы.
- 6 Двумерные массивы.
- 7 Строки.
- 8 Классы и операции.
- 9 Наследование.
- 10 Структуры.

Полученные навыки и знания будут использоваться в дальнейших проектах.

Список использованных источников

- 1 Павловская Т. А., С# Программирование на языке высокого уровня: Практикум. — СПб.: Питер, 2009. — 432 с.: ил. — (Серия «Учебное пособие»).