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 7

Студент группы 6ИСб-1 И. Нозимзода

Преподаватель С.Ю. Александров

2018

Содержание

[Содержание 2](#_Toc533446795)

[Задания 4](#_Toc533446796)

[Введение 5](#_Toc533446797)

[1 Линейные программы 6](#_Toc533446798)

[1.1 Описание программы 6](#_Toc533446799)

[1.2 Текст программы 6](#_Toc533446800)

[1.3 Тестирование программы 7](#_Toc533446801)

[2 Разветвляющиеся вычислительные процессы 8](#_Toc533446802)

[2.1 Описание программы 8](#_Toc533446803)

[2.2 Текст программы 8](#_Toc533446804)

[2.3 Тестирование программы 9](#_Toc533446805)

[3 Организация циклов 10](#_Toc533446806)

[3.1 Описание программы 10](#_Toc533446807)

[3.2 Текст программы 10](#_Toc533446808)

[3.3 Тестирование программы 12](#_Toc533446809)

[4 Простейшие классы 13](#_Toc533446810)

[4.1 Описание программы 13](#_Toc533446811)

[4.2 Текст программы 13](#_Toc533446812)

[4.3 Тестирование программы 17](#_Toc533446813)

[5 Одномерные массивы 19](#_Toc533446814)

[5.1 Описание программы 19](#_Toc533446815)

[5.2 Текст программы 19](#_Toc533446816)

[5.3 Тестирование программы 21](#_Toc533446817)

[6 Двумерные массивы 22](#_Toc533446818)

[6.1 Описание программы 22](#_Toc533446819)

[6.2 Текст программы 22](#_Toc533446820)

[6.3 Тестирование программы 24](#_Toc533446821)

[7 Строки 25](#_Toc533446822)

[7.1 Описание программы 25](#_Toc533446823)

[7.2 Текст программы 25](#_Toc533446824)

[7.3 Тестирование программы 26](#_Toc533446825)

[8 Классы и операции 27](#_Toc533446826)

[8.1 Описание программы 27](#_Toc533446827)

[8.2 Текст программы 27](#_Toc533446828)

[8.3 Тестирование программы 30](#_Toc533446829)

[9 Наследование 32](#_Toc533446830)

[9.1 Описание программы 32](#_Toc533446831)

[9.2 Текст программы 32](#_Toc533446832)

[9.3 Тестирование программы 36](#_Toc533446833)

[10 Структуры 41](#_Toc533446834)

[10.1 Описание программы 41](#_Toc533446835)

[10.2 Текст программы 42](#_Toc533446836)

[10.3 Тестирование программы 43](#_Toc533446837)

[Заключение 44](#_Toc533446838)

[Список использованных источников 45](#_Toc533446839)

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

# Линейные программы

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

## Описание программы

*Написать программу для расчета по двум формулам*



## 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  {  class Task01\_1  {  public double Z1(double x)  {  return Math.Pow(Math.Cos(3.0 / 8.0 \* Math.PI - x / 4.0), 2.0)-  Math.Pow(Math.Cos(11.0 / 8.0 \* Math.PI + x / 4.0), 2.0);;  }  public double Z2(double x)  {  return Math.Sqrt(2.0) \* Math.Sin(x / 2.0)/2;;  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Введите аргумент к функции (x):");  double x = double.Parse(Console.ReadLine());  Task01\_1 t1 = new Task01\_1();  Console.WriteLine("Для аргумента x = {0} функции Z1 и Z2 равны:\n\nZ1(x) = {1}\n Z2(x) ={2}\n", x, t1.Z1(x), t1.Z2(x));  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 1.1.

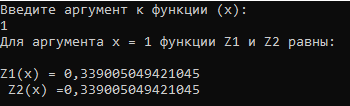


Рисунок 1.1 – Результат работы линейной программы

# Разветвляющиеся вычислительные процессы

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

## Описание программы

*Написать программу, которая определяет, попадает ли точка с заданными координатами в область, закрашенную на рисунке серым цветом (рисунок 2.1). Результат вывести в виде текстового сообщения.*

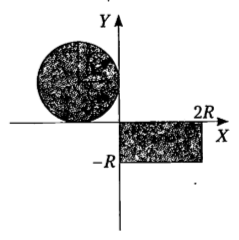


Рисунок 2.1 – График

## Текст программы

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

Листинг 2.1 – Текст файла prog2.cs

|  |
| --- |
| using System;  namespace Lab02  {  class Task02\_2  {  public string O(double x, double y, double R)  {  string flag = "не попадает";  if ((((x + R) \* (x + R) + (y - R) \* (y - R)) <= (R \* R)) || ((x >= -1) && (x <= 2 \* R) && (y >= -R) && (y <= 0))) flag = "попадает";  return flag;  }  }  class Programa  {  static void Main(string[] args)  {  Console.WriteLine("Расположение точки");  Console.WriteLine("Введите X:");  double x = double.Parse(Console.ReadLine());  Console.WriteLine("Введите Y:");  double y = double.Parse(Console.ReadLine());  Console.WriteLine("\n Введите радиус окружности (R):\n");  double R = double.Parse(Console.ReadLine());  Task02\_2 obl = new Task02\_2();  Console.WriteLine("\n Точка {0}", obl.O(x, y, R));  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 2.2.

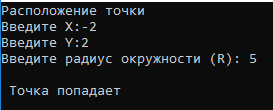


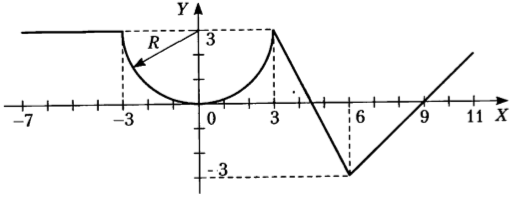
Рисунок 2.2 – Результат работы программы по разветвляющимся вычислительным процессам

# Организация циклов

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

## Описание программы

*Вычислить и вывести на экран значения функции, заданной* графически*, на итервале от до с шагом dx.*



## Текст программы

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

Листинг 3.1 – Текст файла prog3.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace Lab03  {  class Task03\_1  {  public double One(double x)  {  double y = 0;  if (x < -7 || x > 11) { return double.NaN; }  else  {  if (x < -7) y = 0;  if (x >= -7 && x < -3) y = 3;  if (x >= -3 && x < 3) y = -Math.Sqrt(9 - x \* x) + 3;  if (x >= 3 && x < 6) y = -2 \* x + 9;  if (x >= 6 && x < 11) y = x - 9;  if (x >= 11) y = 2;  return y;  }  }  }  class Task03\_2  {  public string Two(double x, double y, double R)  {  string flag = "Не попадает";  if ((((x + R) \* (x + R) + (y - R) \* (y - R)) <= (R \* R))  || ((x >= -1) && (x <= 2 \* R) && (y >= -R) && (y <= 0))) flag = "Попадает";  return flag;  }  }  class Program  {  static void Main(string[] args)  {  Console.WriteLine("Задание 1.\n Введите начальное х(xn), конечное х(xk) и шаг dx:\n");  double xn = double.Parse(Console.ReadLine());  double xk = double.Parse(Console.ReadLine());  double dx = double.Parse(Console.ReadLine());  Task03\_1 func = new Task03\_1();  Console.WriteLine(" Таблица Значений Функции");  Console.WriteLine("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  Console.WriteLine("| x | f(x) |" +  "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  double x = xn;  while (x <= xk)  {  Console.WriteLine(" {0} {1} ", x, func.One(x));  x += dx;  }  Console.WriteLine("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

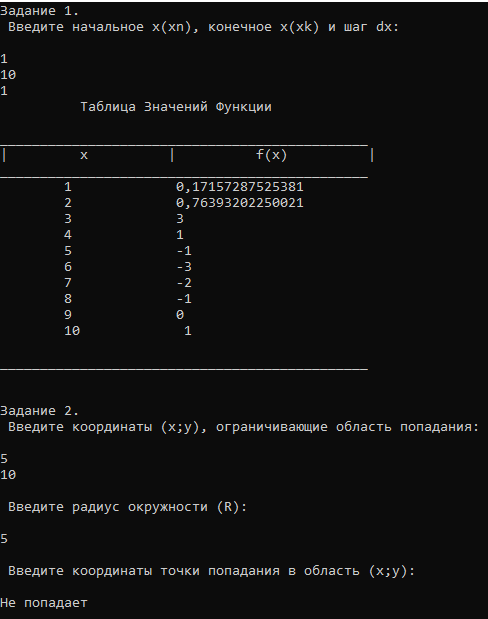


Рисунок 3.1 – Результат работы программы по организации циклов

# 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.

## Описание программы

*Составить описание класса прямоугольника со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольника на плоскости, изменение размеров, построение наименьшего прямоугольника, содержащего 2 заданных прямоугольника и прямоугольника, являющегося общей частью (пересечением) 2-х прямоугольников.*

## Текст программы

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

Листинг 4.1 – Текст файла prog4.cs

|  |
| --- |
| using System;  namespace Lab4  {  public class Rectangle  {  public Rectangle(int x, int y, int width, int height)  {  \_x = x;  \_y = y;  \_width = width;  \_height = height;  }  private int \_x, \_y, \_width, \_height;  public void Relocation(int x, int y)  {  \_x += x;  \_y += y;  }  public void Minimal(int x, int y, int width, int height)  {  int minimalRectX, minimalRectY, minimalWidth, minimalHeight;  if (\_x < x )  minimalRectX = \_x;  else  minimalRectX = x;  if (\_y < y)  minimalRectY = \_y;  else  minimalRectY = y;  if ((Math.Abs(\_x) + \_width) > (Math.Abs(x) + width))  minimalWidth = Math.Abs(\_x) + \_width;  else  minimalWidth = Math.Abs(x) + width;  if ((Math.Abs(\_y) + \_height) > (Math.Abs(y) + height))  minimalHeight = Math.Abs(\_y) + \_height;  else  minimalHeight = Math.Abs(y) + height;  Console.Write(  "Минимальный прямоугольник\n " +  "X: {0}\n" +  "Y: {1}\n" +  "Ширина: {2}\n" +  "Высота: {3}\n",  minimalRectX, minimalRectY, minimalWidth, minimalHeight);  }  public void Сross(int x, int y, int width, int height)  {  double x1 = \_x, x2 = \_x + \_width,  y1 = \_y + \_height, y2 = \_y,  x3 = x, x4 = x + width,  y3 = y + height, y4 = y,  x\_min = \_x, x\_max = \_x,  y\_min = \_y, y\_max = \_y,  x1\_sred = 0, x2\_sred = 0,  y1\_sred = 0, y2\_sred = 0;  double[] A = { x1, x2, x3, x4 }, B = { y1, y2, y3, y4 };  for (int i = 0; i < 4; i++)  {  if (x\_max < A[i])  {  x\_max = A[i];  }  if (y\_max < B[i])  {  y\_max = B[i];  }  if (x\_min > A[i])  {  x\_min = A[i];  }  if (y\_min > B[i])  {  y\_min = B[i];  }  }  for (int i = 0; i < 4; i++)  {  if ((A[i] != x\_max) && (A[i] != x\_min))  {  x1\_sred = A[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((A[i] != x\_max) && (A[i] != x\_min) && (A[i] != x1\_sred))  {  x2\_sred = A[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((B[i] != y\_max) && (B[i] != y\_min))  {  y1\_sred = B[i];  break;  }  }  for (int i = 0; i < 4; i++)  {  if ((B[i] != y\_max) && (B[i] != y\_min) && (B[i] != y1\_sred))  {  y2\_sred = B[i];  break;  }  }  Console.Write(  "Прямоугольник на пересечении\n " +  "X: {0}\n" +  "Y: {1}\n" +  "Ширина: {2}\n" +  "Высота: {3}\n",  x2\_sred, y2\_sred, x1\_sred, y1\_sred);  }  public void Resize(int width, int height)  {  \_width += width;  \_height += height;  }  public void ShowLocationInfo()  {  Console.Write(  "X: {0}\n" +  "Y: {1}\n " +  "Ширина: {2}\n" +  "Высота: {3}\n",  \_x, \_y, \_width, \_height);  }  }    class Lab4  {  static void Main(string[] args)  {  Console.WriteLine("Строим прямоугольник в декартовой системе координат (x и y) по высоте и ширине");  Console.Write("Введите X: ");  int x = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите Y: ");  int y = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите ширину: ");  int width = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите высоту: ");  int height = Convert.ToInt32(Console.ReadLine());  Rectangle rect1 = new Rectangle(x, y, width, height);  do  {  Console.Write("\n");  Console.Write("Варианты команд:\n" +  "1.Перемещение по плоскости\n" +  "2.Изменение размеров\n" +  "3.Показать данные о прямоугольнике\n" +  "4.Минимальный прямоугольник содержащий 2 заданных\n" +  "5.Прямоугольник получившийся на пересечении 2\n" +  "6.Выход\n");  Console.Write("Введите номер пункта меню: ");  switch (Convert.ToInt32(Console.ReadLine()))  {  case 1:  {  Console.Write("На сколько по X: ");  int tempX = Convert.ToInt32(Console.ReadLine());  Console.Write("На сколько по Y: ");  int tempY = Convert.ToInt32(Console.ReadLine());  rect1.Relocation(tempX, tempY);  }  break;  case 2:  {  Console.Write("Изменить ширину на ");  int tempWidth = Convert.ToInt32(Console.ReadLine());  Console.Write("Измнить высоту на ");  int tempHeight = Convert.ToInt32(Console.ReadLine());  rect1.Resize(tempWidth, tempHeight);  }  break;  case 3:  rect1.ShowLocationInfo();  break;  case 4:  {  Console.WriteLine("Данные о 1ом прямоугольнике");  rect1.ShowLocationInfo();  Console.WriteLine("Введите данные о 2ом прямоугольнике");  Console.Write("Введите X: ");  int tempX = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите Y: ");  int tempY = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите ширину: ");  int tempWidth = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите высоту: ");  int tempHeight = Convert.ToInt32(Console.ReadLine());  rect1.Minimal(tempX, tempY, tempWidth, tempHeight);  }  break;  case 5:  {  Console.WriteLine("Данные о 1ом прямоугольнике");  rect1.ShowLocationInfo();  Console.WriteLine("Введите данные о 2ом прямоугольнике");  Console.Write("Введите X: ");  int tempX = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите Y: ");  int tempY = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите ширину: ");  int tempWidth = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите высоту: ");  int tempHeight = Convert.ToInt32(Console.ReadLine());  rect1.Сross(tempX, tempY, tempWidth, tempHeight);  }  break;  case 6:  return;  default:  Console.WriteLine("Не, ну это не серьёзно, вводи!");  break;  }  } while (true);  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунках 4.1, 4.2.

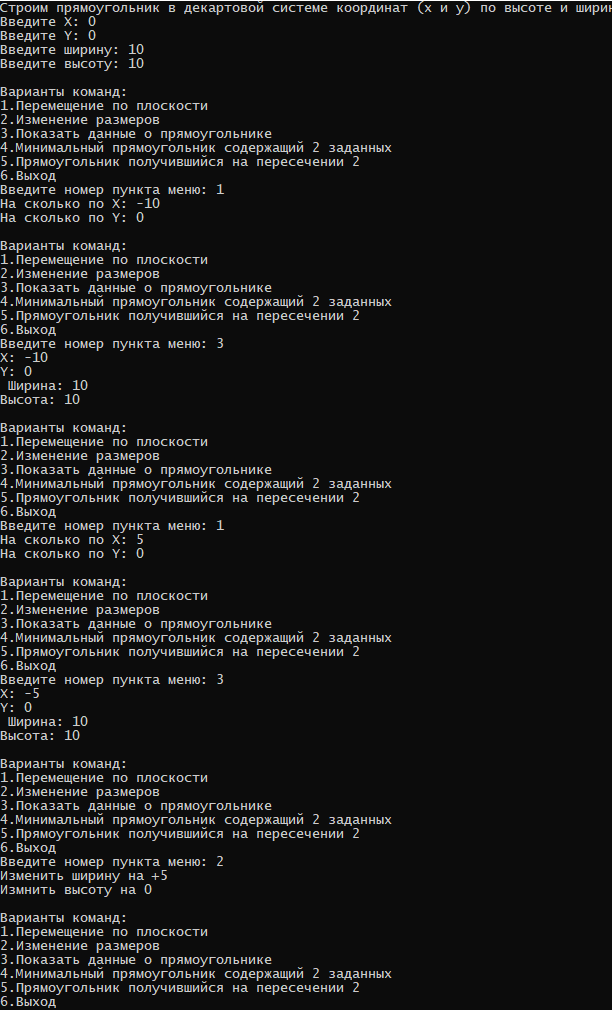


Рисунок 4.1 – Результат работы программы по простейшим классам

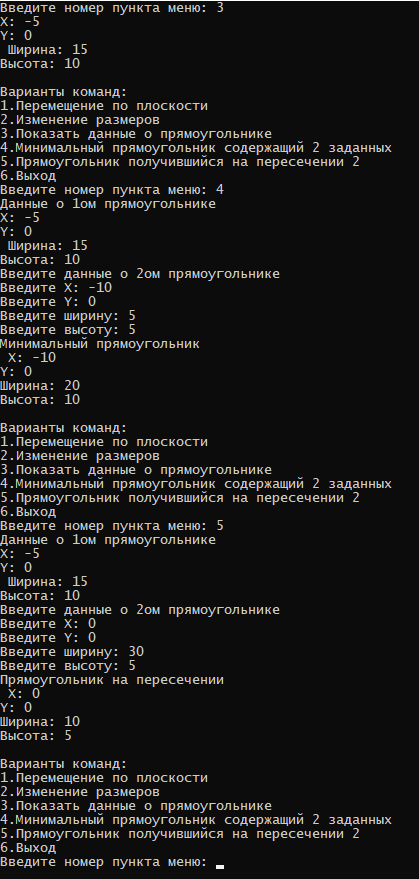


Рисунок 4.2 – Результат работы программы по простейшим классам

# 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.

## Описание программы

*В одномерном массиве, состоящем из n целых элементов, вычислить:*

*1) номер максимального элемента массива;*

*2) произведение элементов массива, расположенных между первым и вторым нулевыми элементами.*

*Преобразовать массив таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях.*

## 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;  namespace Lab5  {  class Lab5  {  static void Main()  {  Console.Write("Введите количество элементов масива: ");  int length = Convert.ToInt32(Console.ReadLine());  List<double> array = new List<double>(length);  for (int i = 0; i < length; i++)  {  Console.Write("Введите значение: ");  array.Add(Convert.ToDouble(Console.ReadLine()));  }  Console.Write("Введенный массив: ");  foreach (var item in array)  {  Console.Write(item + " ");  }  Console.Write("\n");  int maxElIndex = 0;  double maxEl = 0;  for (int i = 0; i < array.Count; i++)  {  if (maxEl < array[i])  {  maxEl = array[i];  maxElIndex = i;  }  }  Console.WriteLine("Индекс максимального элемента: " + maxElIndex);  int zeroCount = 0;  double sum = 1;  foreach (var item in array)  {  if (item == 0)  zeroCount++;  if (zeroCount == 1)  if (item != 0)  sum \*= item;  }  if (zeroCount>0)  Console.WriteLine("Произведение элементов массива, расположенных между первым и вторым нулевыми элементами: " + sum);  else  Console.WriteLine("Нулей меньше двух");  List<double> even = new List<double>();  List<double> notEven = new List<double>();  for (int i = 0; i < array.Count; i++)  {  if ((i % 2) == 0)  {  even.Add(array[i]);  }  else  {  notEven.Add(array[i]);  }  }  array.Clear();  for (int i = 0; i < notEven.Count; i++)  array.Add(notEven[i]);  for (int i = 0; i < even.Count; i++)  array.Add(even[i]);  Console.Write("Отсортированный массив (нечетные в первой половине, четные во второй): ");  foreach (var item in array)  Console.Write(item + " ");  Console.Read();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 5.1.

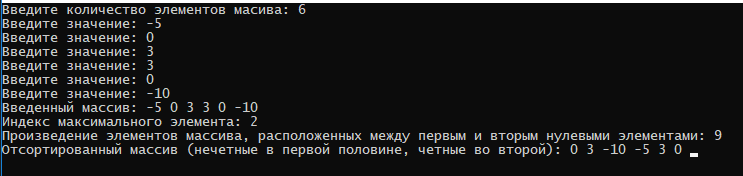


Рисунок 5.1 – Результат работы программы по одномерным массивам

# 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.

## Описание программы

*Для заданной матрицы размером 8 на 8 найти такие k, что k-я строка матрицы совпадает с k-м столбцом. Найти сумму элементов в тех строках, которые содержат хотя бы один отрицательный элемент.*

## Текст программы

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

Листинг 6.1 – Текст файла prog6.cs

|  |
| --- |
| using System;  namespace Lab6  {  class Lab6  {  static void Main(string[] args)  {  int n = 8, m = 8;  int[,] mas = new int[n, m];  Random ran = new Random();  // Инициализируем данный массив  for (int i = 0; i < n; i++)  {  for (int j = 0; j < m; j++)  {  mas[i, j] = ran.Next(-1, 3);  }  }  for (int i = 0; i < n; i++)  {  mas[i, 4] = 0;  }  for (int j = 0; j < m; j++)  {  mas[4, j] = 0;  }  for (int i = 0; i < n; i++)  {  for (int j = 0; j < m; j++)  {  Console.Write("{0}\t", mas[i, j]);  }  Console.WriteLine();  }  bool equal = false;  for (int i = 0; i < n; ++i)  {  equal = true;  for (int j = 0; j < n; ++j)  {  if (mas[i, j] != mas[j, i])  {  equal = false;  break;  }  }  if (equal)  {  Console.WriteLine("k = {0}", i);  }  }  for (int i = 0; i < n; ++i)  {  int sum = 0;  bool negative = false;  for (int j = 0; j < n; ++j)  {  sum += mas[i, j];  if (mas[i, j] < 0) negative = true;  }  if (negative)  {  Console.WriteLine("stroka " + i + ": summa = " + sum);  }  }  Console.Read();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 6.1.

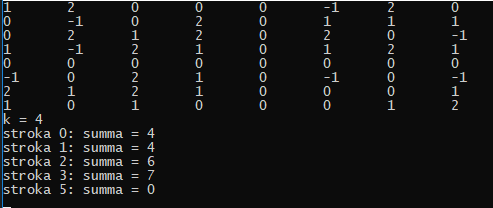


Рисунок 6.1 – Результат работы программы по двумерным массивам

# 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.

## Описание программы

*Написать программу, которая считывает текст из файла и определяет, сколько в нем слов, состоящих из не более чем четырех букв.*

## 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;  namespace Lab7  {  class Lab7  {  static void Main(string[] args)  {  int wordsCount = 0;  StreamReader reader = new StreamReader("my.txt");  string[] words = reader.ReadToEnd().Split(' ');  for (int i = 0; i < words.Length; i++)  if (words[i].Length < 5)  wordsCount++;  Console.WriteLine("Количество слов не более 4 букв: " + wordsCount);  Console.Read();  }  }  } |

## Тестирование программы

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

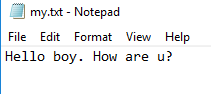


Рисунок 7.1 – Текст из файла \*txt



Рисунок 7.2 - Результат работы программы

# 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.

## Описание программы

*Описать класс «записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по фамилии и доступа к записи по номеру. Написать программу, демонстрирующую разработанные элементы класса.*

## 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;  using System.Linq;  namespace Lab8  {  class Lab8  {  public class Record  {  public int id;  public string Name;  public string Surname;  public string Birthday;  public string PhoneNumber;  public Record(int id, string Name, string Surname, string Birthday, string PhoneNumber)  {  this.id = id;  this.Name = Name;  this.Surname = Surname;  this.Birthday = Birthday;  this.PhoneNumber = PhoneNumber;  }  }  public class Notebook  {  private List<Record> notebookRecords;  public List<Record> NotebookRecords  {  get { return notebookRecords; }  set { notebookRecords = value; }  }  public Notebook()  {  notebookRecords = new List<Record>();  }  public List<Record> SearchRecords(string Name, string Surname, string Birthday, string PhoneNumber)  {  List<Record> ret = new List<Record>();  foreach (Record rec in notebookRecords)  if (rec.Name == Name || rec.Surname == Surname || rec.Birthday == Birthday || rec.PhoneNumber == PhoneNumber)  ret.Add(rec);  return ret;  }  public List<Record> SearchRecords(int id)  {  List<Record> ret = new List<Record>();  foreach (Record rec in notebookRecords)  if (rec.id == id)  ret.Add(rec);  return ret;  }  public void SearchRecords(string Surname)  {  foreach (Record rec in notebookRecords)  if (rec.Surname == Surname)  Console.WriteLine("{0} {1} {2} {3} {4}", rec.id, rec.Name, rec.Surname, rec.Birthday, rec.PhoneNumber);  }  public void AddRecord(int id, string Name, string Surname, string Birthday, string PhoneNumber)  {  Record rec = new Record(id, Name, Surname, Birthday, PhoneNumber);  notebookRecords.Add(rec);  }  public void DeleteRecords(int id)  {  List<Record> records = SearchRecords(id);  foreach (Record rec in records)  notebookRecords.Remove(rec);  }  private static int CompareSurname(Record first, Record second)  {  return first.Surname.CompareTo(second.Surname);  }  public void SortBySurname()  {  notebookRecords.Sort(CompareSurname);  }  public void PrintfInfoByNotebook()  {  foreach (var item in notebookRecords)  {  Console.Write(item.id + "\t");  Console.Write(item.Name + "\t");  Console.Write(item.Surname + "\t");  Console.Write(item.Birthday + "\t");  Console.Write(item.PhoneNumber + "\n");  }  }  }  static void Main(string[] args)  {  Notebook notebook = new Notebook();  while (true)  {  Console.WriteLine("\*\*\*ЗАПИСНАЯ КНИГА\*\*\*");  Console.WriteLine("1. Добавить запись");  Console.WriteLine("2. Удалить запись");  Console.WriteLine("3. Поиск записей");  Console.WriteLine("4. Сортировку по фамилии");  Console.WriteLine("5. Содержание записной книги");  Console.WriteLine("6. Выход");  Console.Write(">");  switch (Convert.ToInt32(Console.ReadLine()))  {  case 1:  {  Console.WriteLine("\*Добавление записи\*");  Console.Write("Введите номер записи: ");  int id = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите имя: ");  string name = Console.ReadLine();  Console.Write("Введите фамилию: ");  string surname = Console.ReadLine();  Console.Write("Введите дату рождения: ");  string birthday = Console.ReadLine();  Console.Write("Введите номер телефона: ");  string phoneNumber = Console.ReadLine();  notebook.AddRecord(id, name, surname, birthday, phoneNumber);  }  break;  case 2:  {  Console.Write("Введите номер удаляемой записи: ");  notebook.DeleteRecords(Convert.ToInt32(Console.ReadLine()));  }  break;  case 3:  {  Console.WriteLine("Введите фамилию");  notebook.SearchRecords(Console.ReadLine());  }  break;  case 4:  notebook.SortBySurname();  break;  case 5:  notebook.PrintfInfoByNotebook();  break;  case 6:  return;  default:  Console.WriteLine("Вводи правильно!");  break;  }  }    }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 8.1,8.2.

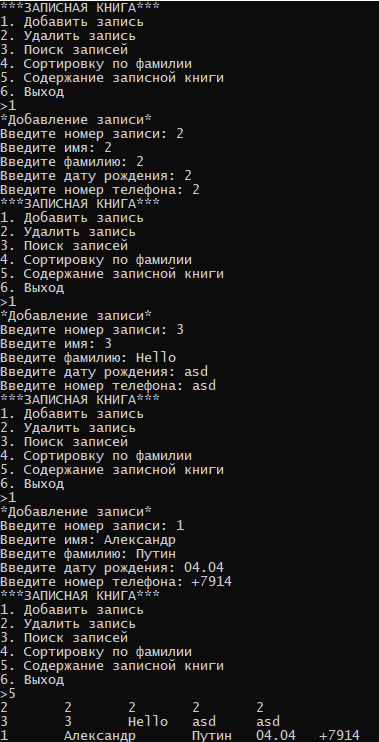


Рисунок 8.1 - Результат работы программы по классам и операциям

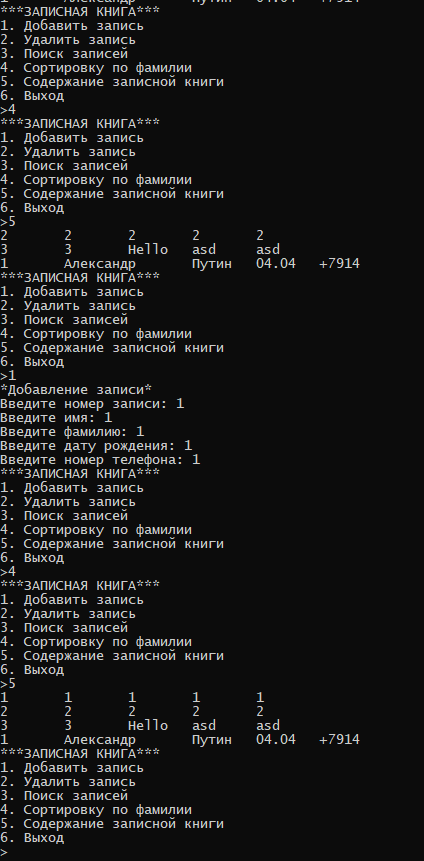


Рисунок 8.2 - Результат работы программы по классам и операциям

# 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.

## 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.*

## Текст программы

Проект состоит из 3 файлов исходного кода, которые приведены в листингах 8.1, 8.2, 8.3.

Листинг 8.1 – 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 8.2 – 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 8.3 – 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();  }  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунках 9.1 - 9.8.

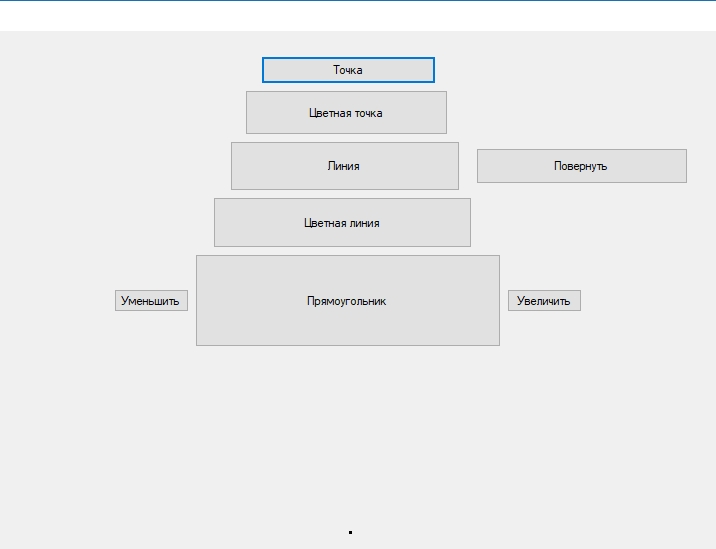


Рисунок 9.1 - Результат работы программы по классу «точка»

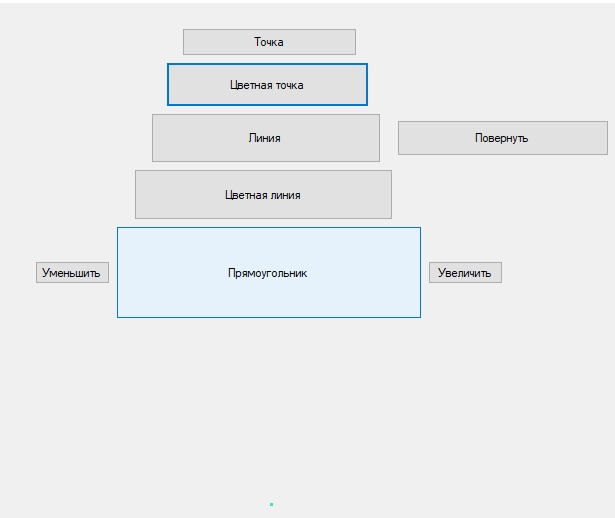


Рисунок 9.2 - Результат работы программы по классу «цветная точка»

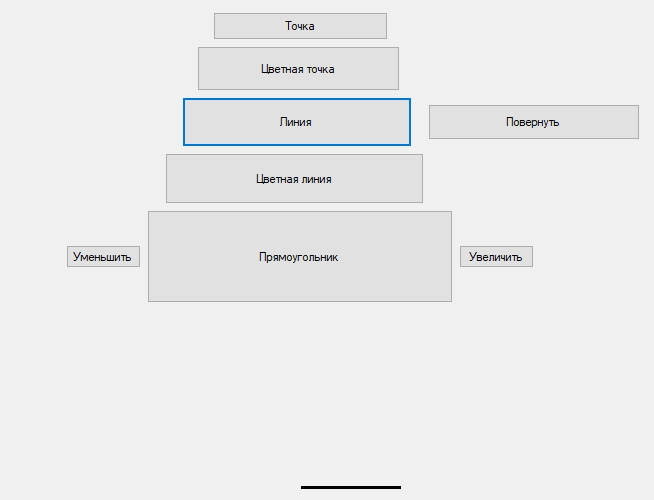


Рисунок 9.3 - Результат работы программы по классу «линия»

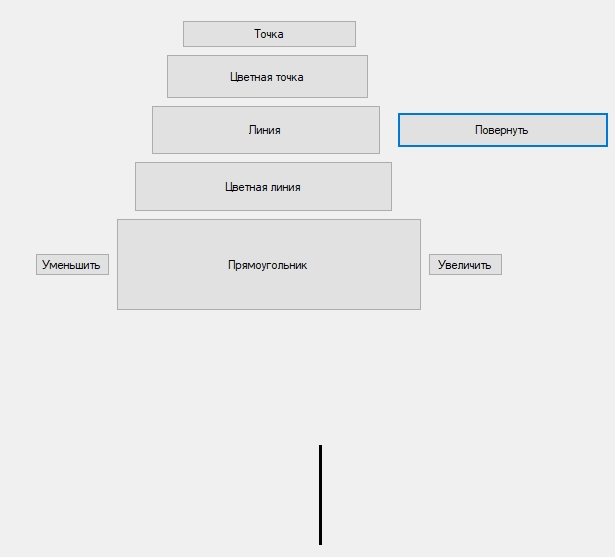


Рисунок 9.4 - Результат работы программы по методу «повернуть линию»

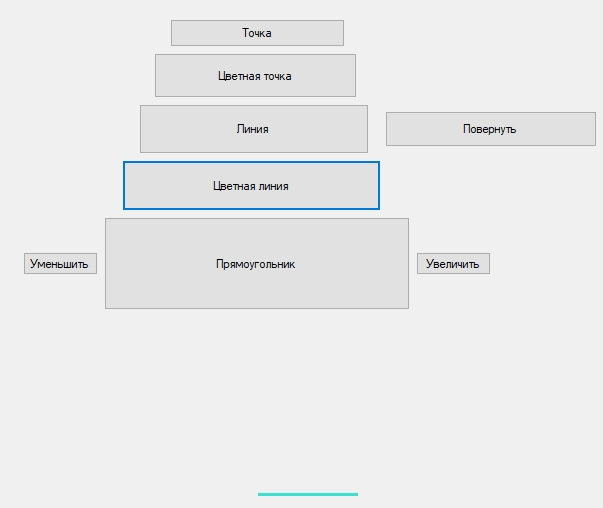


Рисунок 9.5 - Результат работы программы по классу «цветная линия»

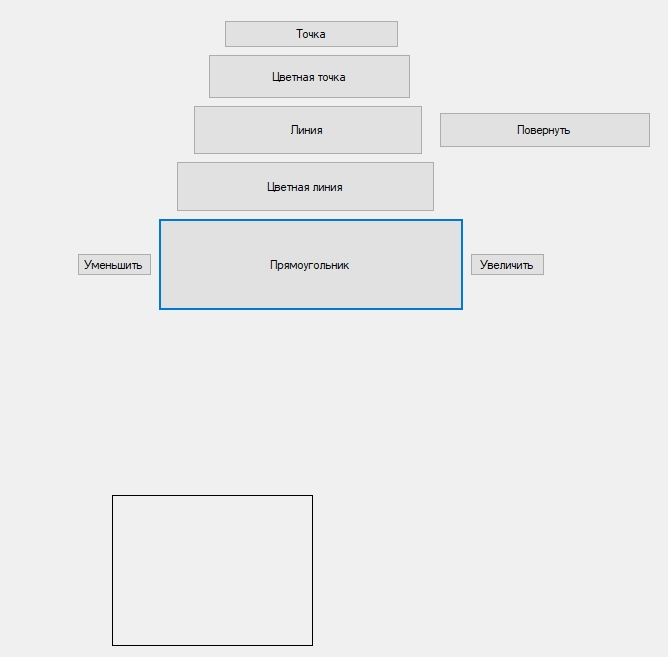


Рисунок 9.6 - Результат работы программы по классу «прямоугольник»

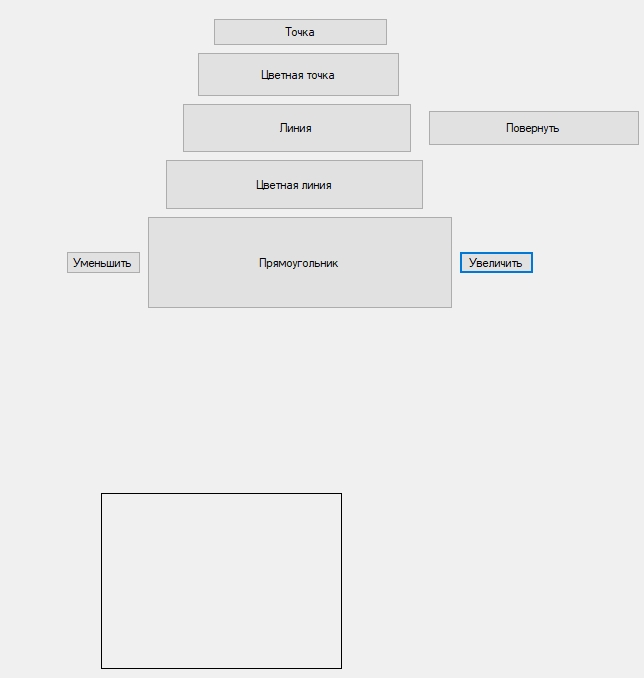


Рисунок 9.7 - Результат работы программы по масштабированию прямоугольника (увеличить)

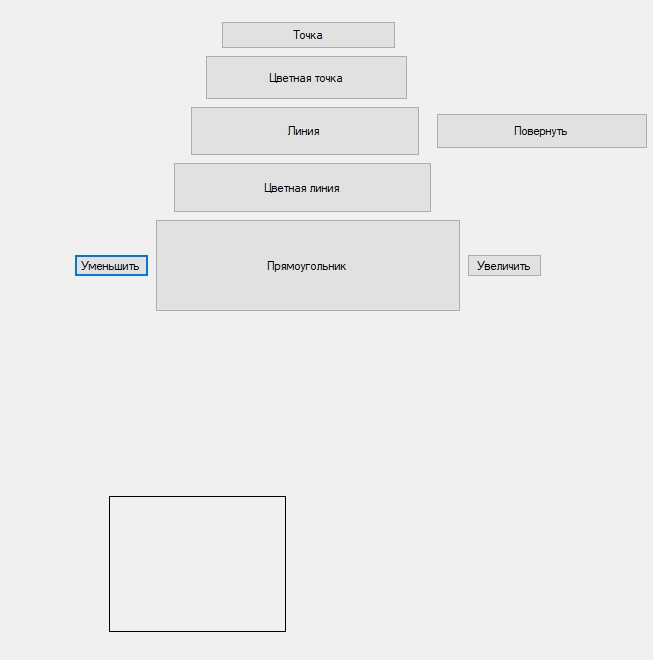


Рисунок 9.8 - Результат работы программы по масштабированию прямоугольника (уменьшить)

# 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.

## Описание программы

*Описать структуру с именем TRAIN, содержащую следующие поля:*

*- название пункта назначения;*

*- номер поезда;*

*- время отправления.*

*Написать программу, выполняющую следующие действия:*

*- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа TRAIN (записи должны быть упорядочены по названиям пунктов назначения по алфавиту);*

*- вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени (если таких поездов 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;  using System.Collections.Generic;  namespace Lab10  {  public struct Train  {  string name;  public int number;  public string date;  public Train(string name, int number, string date)  {  this.name = name;  this.number = number;  this.date = date;  }  public static int Compare(Train first, Train second)  {  return first.name.CompareTo(second.name);  }  public override string ToString()  {  return String.Format("Название пункта назначения " + this.name + " \nНомер поезда " + number + " \nВремя отправления " + date);  }  }  class Lab10  {  static void Main()  {  List<Train> nL = new List<Train>();  for (int i = 0; i < 2; i++)  {  Console.Write("Введите название пункта назначения: ");  string tempName = Console.ReadLine();  Console.Write("Введите номер поезда: ");  int tempNumber = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите время отправления: ");  string tempTime = Console.ReadLine();  nL.Add(new Train(tempName, tempNumber, tempTime));  }  foreach (Train c in nL)  {  Console.WriteLine(c.ToString());  }  Console.WriteLine("\nСортируем по пунктам назначения");  nL.Sort(Train.Compare);  foreach (Train c in nL)  {  Console.WriteLine(c.ToString());  }  Console.WriteLine("Вывод на экран информации о поездах, отправляющихся после введенного с клавиатуры времени");  Console.Write("Введите время ");  string time = Console.ReadLine();  foreach (var item in nL)  {  var temp = time.CompareTo(item.date);  if (temp < 0)  Console.WriteLine(item.ToString());  }  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 10.1.

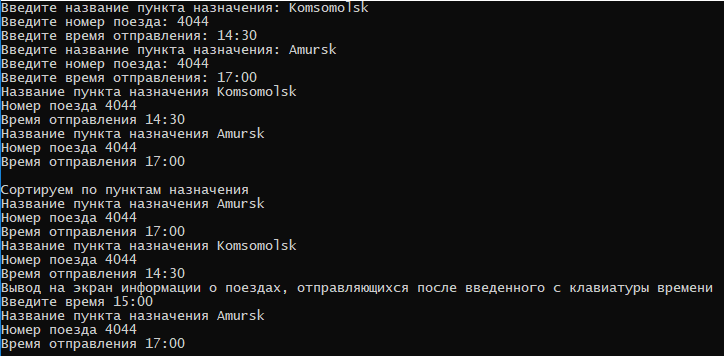


Рисунок 10.1 - Результат работы программы по структурам

Заключение

В ходе изучения дисциплины «Технологии программирования» по изучению языка программирования C# 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., C# Программирование на языке высокого уровня: Практикум. — СПб.: Питер, 2009. — 432 с.: ил. — (Серия «Учебное пособие»).