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 11

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 11](#_Toc533446809)

[4 Простейшие классы 12](#_Toc533446810)

[4.1 Описание программы 12](#_Toc533446811)

[4.2 Текст программы 12](#_Toc533446812)

[4.3 Тестирование программы 14](#_Toc533446813)

[5 Одномерные массивы 15](#_Toc533446814)

[5.1 Описание программы 15](#_Toc533446815)

[5.2 Текст программы 15](#_Toc533446816)

[5.3 Тестирование программы 17](#_Toc533446817)

[6 Двумерные массивы 18](#_Toc533446818)

[6.1 Описание программы 18](#_Toc533446819)

[6.2 Текст программы 18](#_Toc533446820)

[6.3 Тестирование программы 19](#_Toc533446821)

[7 Строки 20](#_Toc533446822)

[7.1 Описание программы 20](#_Toc533446823)

[7.2 Текст программы 20](#_Toc533446824)

[7.3 Тестирование программы 21](#_Toc533446825)

[8 Классы и операции 22](#_Toc533446826)

[8.1 Описание программы 22](#_Toc533446827)

[8.2 Текст программы 22](#_Toc533446828)

[8.3 Тестирование программы 26](#_Toc533446829)

[9 Наследование 27](#_Toc533446830)

[9.1 Описание программы 27](#_Toc533446831)

[9.2 Текст программы 28](#_Toc533446832)

[9.3 Тестирование программы 31](#_Toc533446833)

[10 Структуры 32](#_Toc533446834)

[10.1 Описание программы 32](#_Toc533446835)

[10.2 Текст программы 33](#_Toc533446836)

[10.3 Тестирование программы 34](#_Toc533446837)

[Заключение 35](#_Toc533446838)

[Список использованных источников 36](#_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 . – Текст файла prog1.cs

|  |
| --- |
| using System;  namespace Lab  {  class Lab1  {  public static double Z1(double a)  {  return (1 - 2 \* Math.Pow(Math.Sin(a), 2)) /  (1 + (Math.Sin(2 \* a)));  }  public static double Z2(double a)  {  return (1 - (Math.Tan(a))) /  (1 + (Math.Tan(a)));  }  static void Main(string[] args)  {  Console.Write("Введите A: ");  double a = double.Parse(Console.ReadLine());  Console.WriteLine("Для аргумента x = {0} Функции Z1 и Z2 равны: \n \n Z1(x) = {1} \n Z2(x) = {2} \n", a, Z1(a), Z2(a));  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 1.1.

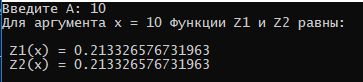


Рисунок . – Результат работы линейной программы

# 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.*

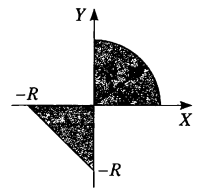


Рисунок 2.1 – График

## Текст программы

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

Листинг . – Текст файла prog2.cs

|  |
| --- |
| using System;  namespace Lab2  {  class Lab2  {  public static string shot(double x, double y, double R)  {  string flag = "не попадает";  if (((Math.Pow(x, 2) + Math.Pow(y, 2) <= (R \* R)) && (y <= x) && (x <= 0)) ||  ((Math.Pow(x, 2) + Math.Pow(y, 2) <= (R \* R)) && (y >= x) && (x >= 0))) flag = "попадает";  return flag;  }  static void Main(string[] args)  {  while (true)  {  Console.Write("Введите радиус окружности (R): ");  double R = double.Parse(Console.ReadLine());  Console.WriteLine("Введите координат точки попадания в область (х;у):");  Console.Write("Введите X: ");  double X = double.Parse(Console.ReadLine());  Console.Write("Введите Y: ");  double Y = double.Parse(Console.ReadLine());  Console.WriteLine("Для координат ({0};{1}) точка в область {2}.", X, Y, shot(X, Y, R));  }  }  }  } |

## 

## Тестирование программы

Результат работы программы приведен на рисунке 2.2.

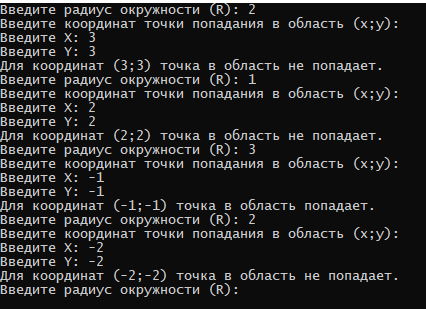


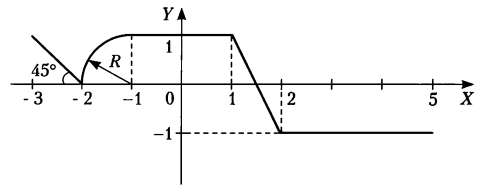
Рисунок .2 – Результат работы программы по разветвляющимся вычислительным процессам

# 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. – Текст файла prog3.cs

|  |
| --- |
| using System;  namespace Lab3  {  class Task03\_1  {  public double One(double x)  {  double y = 0;  if (x < -3 || x > 5) { return double.NaN; }  else  {  if (x < -3) y = 0;  if (x >= -3 && x < -2) y = -x - 2;  if (x >= -2 && x < -1) y = +Math.Sqrt(1 - Math.Pow(x, 2 + 1));  if (x >= -1 && x < 1) y = 1;  if (x >= 1 && x < 2) y = 3 - 2;  if (x >= 2 && x < 5) y = -1;  if (x >= 5) y = 0;  return y;  }  }  class Lab3  {  static void Main(string[] args)  {  Console.WriteLine("Введите X начальное (xn), X конечное (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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  Console.WriteLine("| x | f(x) |" +  "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\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.

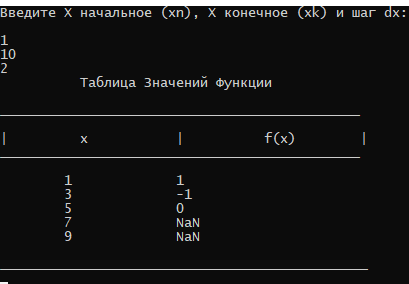


Рисунок 3.1 – Результат работы программы по организации циклов

# 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

*Описать класс, представляющий треугольник. Предусмотреть методы для создания объектов, вычисление площади, периметра. Описать свойства для получения состояния объекта. При невозможности построения треугольника выбрасывается исключение.*

## Текст программы

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

Листинг 4. – Текст файла prog4.cs

|  |
| --- |
| using System;  namespace Lab4  {  class Triangle  {  // Конструктор  public Triangle(double firstSide, double secondSide, double thirdSide)  {  // Проверка все ли стороны положительны  if (firstSide <= 0 || secondSide <= 0 || thirdSide <= 0)  {  throw new IsNotTriangleException("Треугольник со стороной <= 0 не может быть создан");  }  // Проверка являются ли все стороны меньше суммы двух других  if (firstSide + secondSide <= thirdSide  || firstSide + thirdSide <= secondSide  || secondSide + thirdSide <= firstSide)  {  throw new IsNotTriangleException("Одна или более сторон больше чем две другие");  }  // Занесение значений длин сторон в переменные класса  this.\_FirstSide = firstSide;  this.\_SecondSide = secondSide;  this.\_ThirdSide = thirdSide;  }  // Длины сторон  private double \_FirstSide;  private double \_SecondSide;  private double \_ThirdSide;  // Вычисление периметра  public double calcPerimeter()  {  return \_FirstSide + \_SecondSide + \_ThirdSide;  }  // Вычисление площади  public double calcArea()  {  // Вычисление по формуле Герона  double result = 0.25 \* Math.Sqrt((\_FirstSide + \_SecondSide + \_ThirdSide)  \* (\_FirstSide + \_SecondSide - \_ThirdSide)  \* (\_FirstSide + \_ThirdSide - \_SecondSide)  \* (\_SecondSide + \_ThirdSide - \_FirstSide));  return result;  }  // Нахождение точки пересечения медиан  public void calcMediansIntersectionPoint(out double x, out double y)  {  x = 0;  y = 0;  }  // Преобразование объекта в строку ( Этот метод неявно вызывается при выводе объекта на консоль )  override public String ToString()  {  String triangleInfo = "Треугольник со сторонами " + \_FirstSide  + ", " + \_SecondSide + " и " + \_ThirdSide;  return triangleInfo;  }  }  class IsNotTriangleException : Exception  {  public IsNotTriangleException(String message)  {  \_Message = message;  }  private String \_Message;  public override string ToString()  {  return \_Message;  }  }  class Lab4  {  static void Main(string[] args)  {  Random random = new Random();  // Пробуем создать 10 треугольников  for (int counter = 0; counter < 10; counter++)  {  Triangle triangle;  try  {  // Создание треугольника со случайными сторонами ( числа подобраны методом тыка )  triangle = new Triangle(random.Next(3, 10), random.Next(-1, 15), random.Next(3, 10));  // Выводим информацию о треугольнике  Console.WriteLine(triangle);  // Считаем и выводим его периметр  // ( ToString("F") необходима для отображения только двух знаков после запятой )  Console.WriteLine("Периметр треугольника = " + triangle.calcPerimeter().ToString("F"));  // Считаем и выводим его площадь  Console.WriteLine("Площадь треугольника = " + triangle.calcArea().ToString("F"));  // Печатаем пустую строку в качестве разделителя  Console.WriteLine();  }  catch (IsNotTriangleException exception)  {  // Если возникла исключительная ситуация выводим сообщение о ней на консоль  Console.WriteLine("Исключение: " + exception);  // Печатаем пустую строку в качестве разделителя  Console.WriteLine();  }  catch (Exception exception)  {  // Если напечаталась эта строка значит в программе косяк  Console.WriteLine("Неизвестное исключение: " + exception);  // Печатаем пустую строку в качестве разделителя  Console.WriteLine();  }  }  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

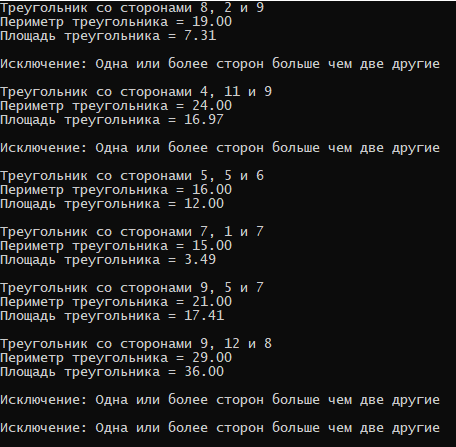


Рисунок 4.1 – Результат работы программы по простейшим классам

# 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 вещественных элементов, вычислить:

- номер минимального по модулю элемента массива;

- сумму модулей элементов массива, расположенных после первого отрицательного элемента.

Сжать массив, удалив из него все элементы, величина которых находится в интервале [a,b]. Освободившийся в конце массива элементы заполнить нулями.

## Текст программы

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

Листинг 5. – Текст файла prog5.cs

|  |
| --- |
| using System;  namespace Lab5  {  class Lab5  {  static void Main(string[] args)  {  Console.Write("Размер массива: ");  int masLenght = Convert.ToInt16(Console.ReadLine());  Random random = new Random();  double[] array = new double[masLenght];  for (int i = 0; i < array.Length; i++)  {  Console.Write("Введите " + (i + 1) + " элемент: ");  array[i] = Convert.ToDouble(Console.ReadLine());  }  Console.Write("Массив: ");  for (int i = 0; i < array.Length; i++)  Console.Write(array[i] + " ");  Console.Write("\n");  int indexMinAbs = 0;  for (int i = 0; i < array.Length; i++)  {  if(Math.Abs(array[indexMinAbs]) > Math.Abs(array[i]))  indexMinAbs = i;  }  Console.WriteLine("Номер минимального по модулю элемента массива: " + indexMinAbs);  double sum = 0;  for (int i = 0; i < array.Length; i++)  {  if (array[i] < 0)  {  for (int j = i + 1; j < array.Length; j++)  {  sum += Math.Abs(array[j]);  }  Console.WriteLine("Сумма модулей элементов массива после первого отрицательного: " + sum);  break;  }  }  Console.Write("A: ");  double a = Convert.ToDouble(Console.ReadLine());  Console.Write("B: ");  double b = Convert.ToDouble(Console.ReadLine());  int iy = 0, ji = iy;  for (; iy < array.Length; iy++)  {  if (array[iy] < a || array[iy] > b)  array[iy - ji] = array[iy];  else  ji++;  }  for (iy = array.Length - ji; iy < array.Length; iy++)  array[iy] = 0;  Console.Write("Массив: ");  for (int i = 0; i < array.Length; i++)  Console.Write(array[i] + " ");  Console.ReadKey();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 5.1.

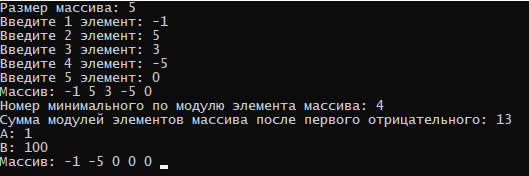


Рисунок 5.1 – Результат работы программы по одномерным массивам

# 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

Коэффициенты системы линейных уравнений заданы в виде прямоугольной матрицы. С помощью допустимых преобразований привести систему к треугольному виду. Найти количество строк, среднее арифметическое элементов которых меньше заданной величины.

## Текст программы

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

Листинг 6. – Текст файла prog6.cs

|  |
| --- |
| using System;  namespace Lab6  {  class Lab6  {  static void ToTriangle(double[,] matrix)  {  double n = matrix.GetLength(0);  for (int i = 0; i < n - 1; i++)  for (int j = i + 1; j < n; j++)  {  double koef = matrix[j, i] / matrix[i, i];  for (int k = i; k < n; k++)  matrix[j, k] -= matrix[i, k] \* koef;  }  }  static void Print(double[,] matrix)  {  for (int i = 0; i < matrix.GetLength(0); i++)  {  for (int j = 0; j < matrix.GetLength(0); j++)  Console.Write("{0:0.0}\t", matrix[i, j]);  Console.WriteLine();  }  Console.WriteLine();  }  static void Main(string[] args)  {  int n = 3;  double[,] matrix = new double[n, n];  var random = new Random();  for (int i = 0; i < matrix.GetLength(0); i++)  for (int j = 0; j < matrix.GetLength(0); j++)  matrix[i, j] = random.Next(1, 9);  Print(matrix);  Console.WriteLine("Задача: найти количество строк, среднее арифметическое элементов которых меньше заданной величины");  Console.Write("Введите среднее арифметическое: ");  double avrArfmt = Convert.ToDouble(Console.ReadLine());  int countLine = 0;  for (int i = 0; i < matrix.GetLength(0); i++)  {  double sum = 0;  for (int j = 0; j < matrix.GetLength(0); j++)  {  sum += matrix[i,j];  }  if ((sum / matrix.GetLength(0)) < avrArfmt) countLine++;  }  Console.WriteLine("Количество строк, удовлетворяющие условию: " + countLine);  Console.WriteLine("Преобразуем матрицу к треугольному виду...");  ToTriangle(matrix);  Print(matrix);  Console.ReadKey();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 6.1.

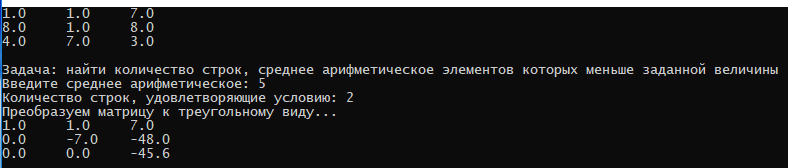


Рисунок 6.1 – Результат работы программы по двумерным массивам

# 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 выводит на экран только строки, не содержащие двухзначных чисел*.*

## Текст программы

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

Листинг 7. – Текст файла prog7.cs

|  |
| --- |
| using System;  using System.IO;  using System.Text.RegularExpressions;  namespace Lab7  {  class Lab7  {  static void Main(string[] args)  {  Console.Write("Текстовый файл (путь): ");  string path = Console.ReadLine();  StreamReader sr = new StreamReader("txt.txt");  string txt = sr.ReadToEnd();  string[] masText = txt.Split('\n');  Regex r = new Regex("[0-9]{2,}");  foreach (var item in masText)  {  if (!(r.IsMatch(item))) Console.WriteLine(item);  }  Console.ReadLine();  }  }  } |

## Тестирование программы

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

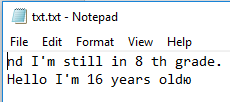


Рисунок 7.1 – Текст из файла \*txt



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, реализующий тип данных «вещественная матрица» и работу с ними. Класс должен реализовать следующие операции над матрицами:

- методы, реализующие проверку типа матрицы (квадратная, диагональная, нулевая, единичная, симметричная, верхняя треугольная, нижняя треугольная);

- операции сравнения на равенство / неравенство;

- доступ к элементу по индексам.

Написать программу, демонстрирующую все разработанные элементы класса.

## Текст программы

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

Листинг 8. – Текст файла prog8.cs

|  |
| --- |
| using System;  namespace Lab8  {  class Lab8  {  static void Main(string[] args)  {  Console.WriteLine("Введите количество строк матрицы");  int n = int.Parse(Console.ReadLine());  Console.WriteLine("Введите количество столбцов матрицы");  int m = int.Parse(Console.ReadLine());  Console.WriteLine("Заполните матрицу");  DoubleMatrix matrix = new DoubleMatrix(n, m);  Random random = new Random();  for (int i = 0; i < matrix.rows; i++)  {  for (int j = 0; j < matrix.cols; j++)  matrix[i, j] = random.Next(-10,10);  }  Console.WriteLine("Матрица :");  for (int i = 0; i < matrix.rows; i++)  {  for (int j = 0; j < matrix.cols; j++)  {  Console.Write(matrix[i, j] + "\t");  }  Console.WriteLine();  }  matrix.Method(); //нулевая или не нулевая  if (matrix.Method1() == false) // проверка на семмитричность  {  Console.WriteLine("Матрица не симметричная");  }  else Console.WriteLine("Матрица симметричная");  matrix.Method2(); // Проверка квадратная  if (matrix.Method3()) // проверка на единичность  {  Console.WriteLine("Матрица единичная");  }  else Console.WriteLine("Матрица не единичная");  if (matrix.Method4() == true) //Проверка на диагональность  {  Console.WriteLine("Матрица диагональная");  }  else Console.WriteLine("Матрица не диагональная");  if (matrix.Method5())  {  Console.WriteLine("Матрица верхняя треугольная ");  }  else Console.WriteLine("Матрица не верхняя треугольная ");  if (matrix.Method6())  {  Console.WriteLine("Матрица нижняя треугольная ");  }  else Console.WriteLine("Матрица не нижняя треугольная ");  Console.WriteLine("Получим элемент по индексу [0,0]: " + matrix[0, 0]); //доступ к элементу по индексам  Console.ReadKey();  }  }  class DoubleMatrix  {  private double[,] matrix;  public int rows, cols;  private int Length;  int r = 0;  bool a = true, t = false;  public DoubleMatrix(int rows, int cols)  {  this.rows = rows;  this.cols = cols;  matrix = new double[this.rows, this.cols];  Length = rows \* cols;  }  public double this[int index1, int index2]  {  get { return matrix[index1, index2]; }  set { matrix[index1, index2] = value; }  }  public void Method() //нулевая или не нулевая  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  if (matrix[i, j] == 0)  { r = r + 1; }  }  }  if (r == cols \* rows)  { Console.WriteLine("Матрица нулевая"); }  else { Console.WriteLine("Матрица не нулевая"); }  }  public bool Method1() // проверка на семмитричность  {  if (cols == rows)  {  for (int i = 0; i < matrix.GetLength(0); ++i)  {  for (int j = 0; j < matrix.GetLength(1); ++j)  if (matrix[i, j] != matrix[j, i])  {  a = false;  break;  }  if (!a) break;  }  return a;  }  else return false;  }  public void Method2() // Проверка квадратная  {  if (rows == cols)  {  Console.WriteLine("Матрица квадратная");  }  else Console.WriteLine("Матрица не квадратная");  }  public bool Method3() // проверка на единичность  {  if (rows == cols)  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  if (i == j && matrix[i, j] != 1)  {  return false;  }  else if (i != j && matrix[i, j] != 0)  { return false; }  }  }  return true;  }  return false;  }  public bool Method4() //Проверка на диагональность  {  if (rows == cols)  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  if (i != j)  {  if (matrix[i, j] == 0)  {  t = true;  }  else t = false;  break;  }  }  }  }  return t;  }  public bool Method5()  {  if (rows == cols)  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  if (matrix[i, j] != 0 && i > j)  {  return false;  }  }  }  return true;  }  return false;  }  public bool Method6()  {  if (rows == cols)  {  for (int i = 0; i < rows; i++)  {  for (int j = 0; j < cols; j++)  {  if (matrix[i, j] != 0 && i < j)  {  return false;  }  }  }  return true;  }  return false;  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 8.1.

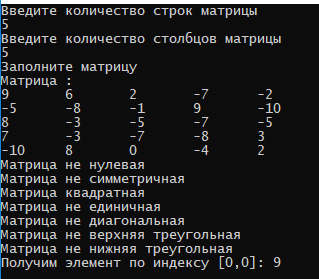


Рисунок 8.1 - Результат работы программы по классам и операциям

# 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

Описать базовый класс «Строка».

Обязательные поля класса:

- поле для хранения символов строки;

- значение типа word для хранения длины строки в байтах.

Реализовать обязательные методы следующего назначения:

- конструктор без параметров;

- конструктор,принимающий в качестве параметра символ;

- конструктор,принимающий в качестве параметра строковый литерал;

- метод получения длины строки;

- метод очистки строки (сделать строку пустой);

Описать производный от класса «Строка» класс «Битовая\_Строка». Строки данного класса могут содержать только символы '0' или '1'. Если в составе инициализирующей строки будут встречены любые символы, отличные от допустимых, класс «Битовая\_строка» принимает нулевое значение. Содержимое данных строк рассматривается как двоичное число. Отрицательные числа хранятся в дополнительном коде.

Для класса «Битовая\_строка» описать следующие методы:

- конструктор, принимающий в качестве параметра строковый литерал;

- деструктор;

- изменение знака на противоположный (перевод числа в дополнительный код);

- присваивание;

- вычисление арифметической суммы строк;

- проверка на равенство;

В случае необходимости более коротка битовая строка расширяется влево знаковым разрядом.

## Текст программы

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

Листинг 8. – Текст файла prog9.cs

|  |
| --- |
| using System;  namespace Lab9  {  class Lab9  {  private class String  {  private int \_length;  private string \_str;  // конструктор без параметров  public String()  {  }  // конструктор, принимающий в качестве параметра строковый литерал  public String(string str)  {  \_str = str;  \_length = str.Length;  }  // конструктор, принимающий в качестве параметра символ;  public String(char ch)  {  \_str = Convert.ToString(ch);  \_length = 1;  }  // Метод возвращающий длинну строки.  public int GetLength()  {  return \_length;  }  // Метод очищающий стоку.  public void Clear()  {  \_str = "";  \_length = 0;  }  public override string ToString()  {  return \_str;  }  }  private class BitString : String  {  public bool znak;  private string \_str;  private int \_length;  // конструктор, принимающий в качестве параметра строковый литерал  public BitString(string str)  {  \_str = str;  \_length = str.Length;  }  // деструктор  ~BitString()  {  }  public override string ToString()  {  return \_str;  }  public static BitString operator +(BitString m1, BitString m2)  {  BitString str = new BitString("0000000000000000000");  char[] a = str.\_str.ToCharArray();  for (int i = m1.\_str.Length - 1; i >= 0; i--)  a[i] = Convert.ToString(Convert.ToInt32(Convert.ToString(m1.\_str[i])) + Convert.ToInt32(Convert.ToString(m2.\_str[i])))[0];  for (int i = m1.\_str.Length - 1; i > 0; i--)  {  if (a[i] == '2')  {  a[i - 1] = Convert.ToString(Convert.ToInt32(Convert.ToString(a[i - 1])) + 1)[0];  a[i] = '0';  }  if (a[i] == '3')  {  a[i - 1] = Convert.ToString(Convert.ToInt32(Convert.ToString(a[i - 1])) + 1)[0];  a[i] = '1';  }  }  string g = "";  for (int i = 0; i < a.Length; i++)  g += a[i];  str.\_str = g;  return str;  }  public static bool operator ==(BitString m1, BitString m2)  {  bool x;  if (m1.\_str == m2.\_str)  x = true;  else x = false;  return x;  }  public static bool operator !=(BitString m1, BitString m2)  {  bool x;  if (m1.\_str != m2.\_str)  x = true;  else x = false;  return x;  }  public static BitString Dop\_kod(BitString m1)  {  char[] a = m1.\_str.ToCharArray();  if (m1.znak == false)  {  for (int i = a.Length - 1; i >= 0; i--)  {  if (a[i] == '0')  a[i] = '1';  else  a[i] = '0';  }  a[0] = Convert.ToChar(Convert.ToInt32(a[a.Length - 1]) + 1);  for (int i = a.Length - 1; i >= 0; i--)  {  if (a[i] == '2')  {  a[i - 1] = Convert.ToChar(Convert.ToInt32(a[i]) + 1);  a[i] = '0';  }  if (a[i] == '3')  {  a[i - 1] = Convert.ToChar(Convert.ToInt32(a[i]) + 1);  a[i] = '1';  }  }  m1.znak = true;  }  string g = "";  for (int i = 0; i < a.Length; i++)  g += a[i];  m1.\_str = g;  return m1;  }  public static BitString Prisvaivanie(string str)  {  BitString m1 = new BitString(str);  return m1;  }  }  static void Main(string[] args)  {  BitString m1, m2, m3;  m1 = BitString.Prisvaivanie("0000000000000110001");  Console.WriteLine("Битовая строка 1: " + m1.ToString());  m2 = BitString.Prisvaivanie("0000000000000011001");  Console.WriteLine("Битовая строка 2: " + m2.ToString());  Console.Write("\n");  if (m2 == m1)  Console.WriteLine("Строки 1 и 2 равны");  else  Console.WriteLine("Строки 1 и 2 не равны");  Console.Write("\n");  Console.WriteLine("Строка 1 меняет знак на противоположный...");  m1 = BitString.Dop\_kod(m1);  Console.WriteLine("Строкa 1: " + m1.ToString());  Console.Write("\n");  m3 = m1 + m2;  Console.WriteLine("Складываем 1 и 2 битовые стоки... Ответ: " + m3.ToString());    Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 9.1.

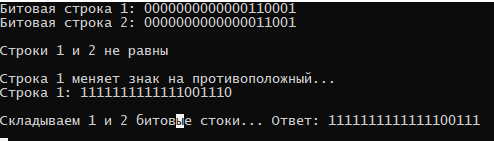


Рисунок 9.1 - Результат работы программы по классу «Битовая строка»

# 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 MARSH, содержащую следующие поля:*

*- название начального пункта маршрута;*

*- название конечного пункта маршрута;*

*- номер маршрута.*

*Написать программу, выполняющую следующие действия:*

*- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа MARSH (записи должны быть упорядочены по номерам маршрутов);*

*- вывод на экран информации о маршруте, номер которого введен с клавиатуры (если таких маршрутов нет, вывести соответствующее сообщение).*

## Текст программы

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

Листинг 10. – Текст файла prog10.cs

|  |
| --- |
| using System;  namespace Lab10  {  struct MARSH  {  public string nachalniy\_punkt\_marshryta;  public string konechniy\_punkt\_marshryta;  public int nomer\_marshryta;  public override string ToString()  {  return (string.Format(@"  Начальный пункт назначения: {0}  Конечный пункт назначения: {1}  Номер маршрута: {2}", nachalniy\_punkt\_marshryta, konechniy\_punkt\_marshryta, nomer\_marshryta));  }  }  class Lab10  {  static void Main(string[] args)  {  int n = 5;  MARSH[] mas = new MARSH[n];  for (int i = 0; i < n; i++)  {  Console.Write("Начальный пункт маршрута: ");  mas[i].nachalniy\_punkt\_marshryta = Convert.ToString(Console.ReadLine());  Console.Write("Конечный пункт маршрута: ");  mas[i].konechniy\_punkt\_marshryta = Convert.ToString(Console.ReadLine());  Console.Write("Номер маршрута: ");  mas[i].nomer\_marshryta = Convert.ToInt32(Console.ReadLine());  Console.Write("\n");  }  Console.WriteLine("Сортировка по номерам маршрутов...");  for (int i = 0; i < n - 1; i++)  {  for (int j = i + 1; j < n; j++)  {  if (mas[i].nomer\_marshryta > mas[j].nomer\_marshryta)  {  MARSH x = mas[i];  mas[i] = mas[j];  mas[j] = x;  }  }  }  Console.WriteLine("Информация в базе:");  Console.WriteLine();  for (int i = 0; i < n; i++)  Console.WriteLine(mas[i]);  Console.WriteLine("Введите номер маршрута для вывода информации: ");  int fam = Convert.ToInt32(Console.ReadLine());  int k = 0;  for (int i = 0; i < n; i++)  {  if (mas[i].nomer\_marshryta == fam)  {  Console.WriteLine(mas[i]);  k++;  }  } if (k == 0) Console.WriteLine("Таких маршрутов нет!");  Console.ReadKey();  } }} |

## Тестирование программы

Результат работы программы приведен на рисунке 10.1.

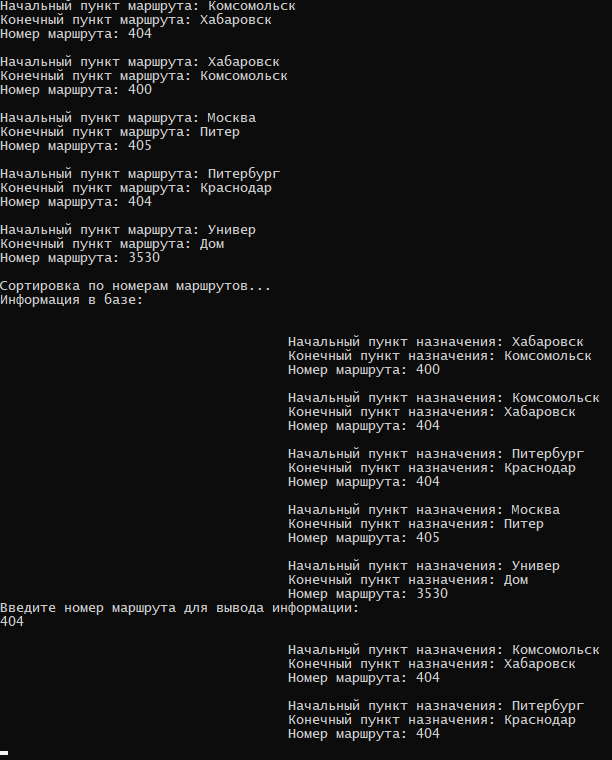


Рисунок 10.1 - Результат работы программы по структурам

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»).