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 12

Студент группы 6ИСб-1 О. Хусейнов

Преподаватель С.Ю. Александров

2018

Содержание

[Содержание 2](#_Toc533447954)

[Задания 4](#_Toc533447955)

[Введение 5](#_Toc533447956)

[1 Линейные программы 6](#_Toc533447957)

[1.1 Описание программы 6](#_Toc533447958)

[1.2 Текст программы 6](#_Toc533447959)

[1.3 Тестирование программы 7](#_Toc533447960)

[2 Разветвляющиеся вычислительные процессы 8](#_Toc533447961)

[2.1 Описание программы 8](#_Toc533447962)

[2.2 Текст программы 8](#_Toc533447963)

[2.3 Тестирование программы 9](#_Toc533447964)

[3 Организация циклов 10](#_Toc533447965)

[3.1 Описание программы 10](#_Toc533447966)

[3.2 Текст программы 10](#_Toc533447967)

[3.3 Тестирование программы 12](#_Toc533447968)

[4 Простейшие классы 13](#_Toc533447969)

[4.1 Описание программы 13](#_Toc533447970)

[4.2 Текст программы 13](#_Toc533447971)

[4.3 Тестирование программы 15](#_Toc533447972)

[5 Одномерные массивы 16](#_Toc533447973)

[5.1 Описание программы 16](#_Toc533447974)

[5.2 Текст программы 16](#_Toc533447975)

[5.3 Тестирование программы 18](#_Toc533447976)

[6 Двумерные массивы 19](#_Toc533447977)

[6.1 Описание программы 19](#_Toc533447978)

[6.2 Текст программы 19](#_Toc533447979)

[6.3 Тестирование программы 21](#_Toc533447980)

[7 Строки 22](#_Toc533447981)

[7.1 Описание программы 22](#_Toc533447982)

[7.2 Текст программы 22](#_Toc533447983)

[7.3 Тестирование программы 23](#_Toc533447984)

[8 Классы и операции 24](#_Toc533447985)

[8.1 Описание программы 24](#_Toc533447986)

[8.2 Текст программы 24](#_Toc533447987)

[8.3 Тестирование программы 30](#_Toc533447988)

[9 Наследование 31](#_Toc533447989)

[9.1 Описание программы 31](#_Toc533447990)

[9.2 Текст программы 31](#_Toc533447991)

[9.3 Тестирование программы 34](#_Toc533447992)

[10 Структуры 35](#_Toc533447993)

[10.1 Описание программы 35](#_Toc533447994)

[10.2 Текст программы 36](#_Toc533447995)

[10.3 Тестирование программы 37](#_Toc533447996)

[Заключение 38](#_Toc533447997)

[Список использованных источников 39](#_Toc533447998)

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 Lab1  {  class Lab1  {  static void Main(string[] args)  {  Console.Write("Введите a: ");  double a = Convert.ToDouble(Console.ReadLine());  double Z1 = Math.Sin(4.0 \* a) / (1 + Math.Cos(4.0 \* a)) \* (Math.Cos(2.0 \* a) / (1.0 + Math.Cos(2.0 \* a)));  double Z2 = 1.0 / Math.Tan(3.0 / 2.0 \* Math.PI - a);  //double Z3 = Math.Cos(3.0 / 2.0 \* Math.PI - a) / Math.Sin(3.0 / 2.0 \* Math.PI - a);  Console.Write("Z1 = {0}\nZ2 = {1}", Z1, Z2);  Console.Read();  }  }  } |

## 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.*

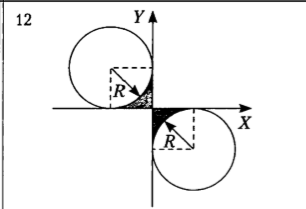


Рисунок 2.1 – График

## 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  {  static bool IsHit(double r, double x, double y)  {  if (((((x >= -r) && (x <= 0)) && ((y >= 0) && (y <= r) && (Math.Sqrt(Math.Pow((x + r), 2) + Math.Pow((y - r), 2)) >= r))) | (((x >= 0) && (x <= r)) && ((y >= -r) && (y <= 0)))) && (Math.Sqrt(Math.Pow((x + r), 2) + Math.Pow((y - r), 2)) >= r))  {  return true;  }  else return false;  }  static void Main()  {  Console.Write("Введите радиус: ");  double r = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите X: ");  double x = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите Y: ");  double y = Convert.ToDouble(Console.ReadLine());  string ans = IsHit(r, x, y) ? "Попадает" : "Не попадает";  Console.WriteLine(ans);  Console.Read();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 2.2.

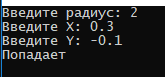


Рисунок .2 – Результат работы программы по разветвляющимся вычислительным процессам

# 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 с помощью ряда Тейлора, на интервале от до с шагом dx с точностью ε.*



## 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 Lab3  {  public static int Fact(Int32 n)  {  int res = 1;  if (n <= 1) res = 1;  else  {  for (int i = n; i > 1; i--)  res \*= i;  }  return res;  }  static void Main(string[] args)  {  Console.Clear();  int c = 0;  double y = 0;  double x1 = 0;  double x2 = 0;  double dx = 0;  double e = 0;  double d = 0;  int f = 1;  int l = 0;  bool u = true;  double bufy = 0;  bool o = true;  while (o)  {  try  {  o = false;  Console.Write("Введите х начальное ");  x1 = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите х конечное ");  x2 = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите шаг ");  dx = Convert.ToDouble(Console.ReadLine());  Console.Write("Введите точность ");  e = Convert.ToDouble(Console.ReadLine());  }  catch  {  o = true;  Console.Clear();  Console.WriteLine("Ошибка ввода");  }  }  Console.WriteLine("Таблица значений");  Console.WriteLine("---------------------------");  Console.WriteLine(" x | y | k ");  Console.WriteLine("---------------------------");  c = 0;  l = 1;  y = 0;  d = 0;  f = 1;  u = true;  for (int i = 1; x1 <= x2; i++)  {  for (int n = 0; u; n++)  {  f = Fact(n);  bufy = (l \* (Math.Pow(x1, 2 \* n))) / f;  y = y + bufy;  if (Math.Abs(bufy) < e) u = false;  d = bufy;  c++;  //f = f \* c;  l = l \* -1;  y = Math.Round(y, 5);  }  Console.Write("|{0,5} ", x1);  Console.Write("|{0,13}|", y);  Console.WriteLine(" {0,3}", c);  x1 = x1 + dx;  }  Console.ReadLine();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 3.1.

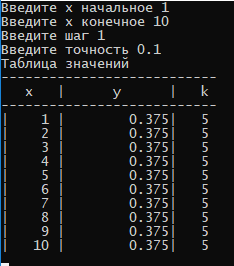


Рисунок 3.1 – Результат работы программы по организации циклов

# 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 круг. Предусмотреть методы для создания объектов, вычисления площади круга, длины окружности и проверки попадания заданной точки внутрь круга. Описать свойства для получения состояния объекта. Написать программу, демонстрирующую все разработанные элементы класса.*

## 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  {  public static class Lab4  {  private static void Main()  {  Circle circle = new Circle(1, new Point(0, 0));  Console.WriteLine(circle);  Console.WriteLine("Проверка на принадлежность точки (0.1,0.1) кругу.");  Console.WriteLine(circle.IsInside(new Point(0.1, 0.1)));  Console.WriteLine("Площадь.");  Console.WriteLine(circle.Square);  Console.WriteLine("Длина окружности.");  Console.WriteLine(circle.Circuit);  Console.ReadKey();  }  }  public class Circle  {  /// <summary>  /// Радиус.  /// </summary>  public double Radius { get; set; }  /// <summary>  /// Положение центра.  /// </summary>  public Point Center { get; set; }  public Circle(double radius, Point center)  {  Radius = radius;  Center = center;  }  /// <summary>  /// Площадь круга.  /// </summary>  public double Square  {  get { return Math.PI \* Radius \* Radius; }  }  /// <summary>  /// Длина окружности.  /// </summary>  public double Circuit  {  get { return Math.PI \* 2 \* Radius; }  }  /// <summary>  /// Проверка на принадлежность точки кругу.  /// </summary>  public bool IsInside(Point point)  {  Point vector = new Point(point.X - Center.X, point.Y - Center.Y);  double distance = vector.X \* vector.X + vector.Y \* vector.Y;  return distance <= Radius \* Radius;  }  public override string ToString()  {  return String.Format("Radius: {0}; Center: {1};", Radius, Center);  }  }  public class Point  {  public Point(double x, double y)  {  X = x;  Y = y;  }  public double X { get; private set; }  public double Y { get; private set; }  public override string ToString()  {  return string.Format("({0}, {1})", X, Y);  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 4.1.

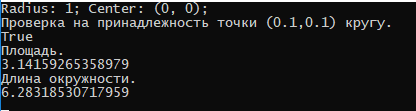


Рисунок 4.1 – Результат работы программы по простейшим классам

# 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 состоящим из n вещественных элементов, вычислить:*

*- номер максимального по модулю элемента массива;*

*- сумму элементов массива, расположенных после первого положительного элемента.*

*Преобразовать массив таким образом, чтобы сначала располагались все элементы, целая часть которых лежит в интервале [a,b], а потом - все остальные.*

## Текст программы

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

Листинг 5. – Текст файла prog5.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  namespace Lab5  {  class Lab5  {  static void Main(string[] args)  {  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;  double sumElAfterOne = 0;  for (int i = 0, startSum = 0; i < array.Count; i++)  {  if (maxEl < array[i])  {  maxEl = array[i];  maxElIndex = i;  }  if (startSum == 1)  sumElAfterOne += array[i];  if (array[i] > 0)  startSum = 1;  }  Console.WriteLine("Индекс максимального элемента: " + maxElIndex);  Console.WriteLine("Cумма элементов массива, расположенных после первого положительного элемента: " + sumElAfterOne);  Console.Write("Введите A: ");  int A = Convert.ToInt32(Console.ReadLine());  Console.Write("Введите B: ");  int B = Convert.ToInt32(Console.ReadLine());  List<double> fromAtoB = new List<double>();  List<double> outAtoB = new List<double>();  for (int i = 0; i < array.Count; i++)  {  if ((int)array[i]>= A && (int)array[i] <= B)  {  fromAtoB.Add(array[i]);  }  else  {  outAtoB.Add(array[i]);  }  }  array.Clear();  for (int i = 0; i < fromAtoB.Count; i++)  array.Add(fromAtoB[i]);  for (int i = 0; i < outAtoB.Count; i++)  array.Add(outAtoB[i]);  Console.Write("Отсортированный массив: ");  foreach (var item in array)  Console.Write(item + " ");  Console.Read();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 5.1.

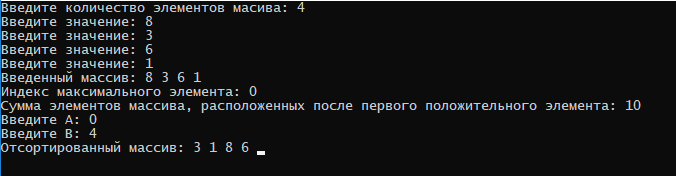


Рисунок 5.1 – Результат работы программы по одномерным массивам

# 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

*Уплотнить заданную матрицу, удаляя из неё столбцы, заполненные нулями. Найти номер первой из строк, содержащих хотя бы один положительный элемент.*

## Текст программы

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

Листинг 6. – Текст файла prog6.cs

|  |
| --- |
| using System;  using System.Collections.Generic;  namespace Lab6  {  class Lab6  {  static void Main(string[] args)  {  int[,] mas = {  {-3,-5,0,-8,-1,0},  {2,1,0,-9,-5,0},  {0,0,0,0,0,0},  {7,8,0,-3,3,0},  {0,0,0,0,0,0},  {9,1,0,-9,-5,0},  {0,0,0,0,0,0},  };  // получаем кол-во строк  int length1 = mas.GetLength(0);  // получаем кол-во столбцов  int length2 = mas.GetLength(1);  //объявляем коллекции для хранения нулевых строк и столбцов  List<int> ls1 = new List<int>();  List<int> ls2 = new List<int>();  Console.WriteLine("======Исходная матрица=========");  // выводим матрицу на консоль и находим нулевые строки  for (int i = 0; i < length1; ++i)  {  bool b = false;  for (int j = 0; j < length2; ++j)  {  if (mas[i, j] >= 0)  Console.Write(" " + mas[i, j]);  else Console.Write(" " + mas[i, j]);  if (mas[i, j] != 0) b = true;  }  if (!b) ls1.Add(i);  Console.WriteLine();  }  // находим нулевые столбцы  for (int i = 0; i < length2; ++i)  {  bool b = false;  for (int j = 0; j < length1; ++j)  {  if (mas[j, i] != 0) b = true;  }  if (!b) ls2.Add(i);  }  // Удаляем нулевые строки и столбцы и находим номер первой из строк (начиная с нуля),  // содержащих хотя бы один положительный элемент  bool B = false; int? Istr = null;  Console.WriteLine("======Удаляем нулевые строки и столбцы=========");  for (int i = 0; i < length1; ++i)  {  if (!ls1.Contains(i))  {  for (int j = 0; j < length2; ++j)  {  if (!ls2.Contains(j))  {  if (mas[i, j] >= 0)  {  if (!B)  {  Istr = i;  B = true;  }  Console.Write(" " + mas[i, j]);  }  else Console.Write(" " + mas[i, j]);  }  }  }  if (!ls1.Contains(i))  Console.WriteLine();  }  Console.WriteLine("Hомер первой из строк (начиная с нуля), содержащих хотя бы один положительный элемент -> {0}", Istr);  Console.ReadKey();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 6.1.

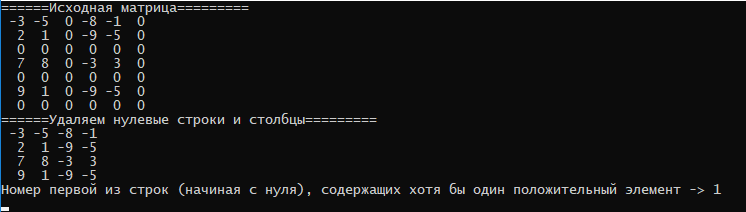


Рисунок 6.1 – Результат работы программы по двумерным массивам

# 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 предложения, начинающиеся с тире, перед которым могут находиться только пробельные символы.*

## 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;  namespace Lab7  {  class Lab7  {  static void Main(string[] args)  {  string[] text = File.ReadAllText("text.txt").Split('.','!','?');  foreach (string item in text)  if(item.Trim()[0] == '-')  Console.WriteLine(item.Trim());  Console.Read();  }  }  } |

## Тестирование программы

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

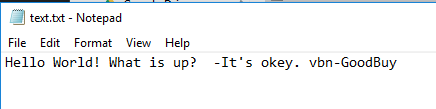


Рисунок 7.1 – Текст из файла \*txt

E:\ТП\Lab\Lab7\Screenshot_1.png

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 «множество», позволяющий выполнять основные операции: добавление и удаление элемента, пересечение, объединение и разность множеств. 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;  using System.Collections.Generic;  using System.Collections;  using System.Linq;  public class Set<T> : ISet<T>  {  T[] array;  public Set()  {  array = new T[0];  }  bool ISet<T>.Add(T item)  {  return Add(item);  }  void ICollection<T>.Add(T item)  {  Array.Resize(ref array, array.Length + 1);  array[array.Length - 1] = item;  }  public bool Add(T item)  {  if (Contains(item))  return false;  Array.Resize(ref array, array.Length + 1);  array[array.Length - 1] = item;  return true;  }  public void Clear()  {  array = new T[0];  }  public bool Contains(T item)  {  for (int i = 0; i < array.Length; i++)  if (array[i].Equals(item))  return true;  return false;  }  public void CopyTo(T[] \_array, int arrayIndex)  {  for (int i = 0; i < array.Length; i++)  \_array[i + arrayIndex] = array[i];  }  public void ExceptWith(IEnumerable<T> other)  {  foreach (T item in other)  {  if (Contains(item))  {  int index = 0;  while (!array[index].Equals(item))  index++;  for (int i = index; i < array.Length - 1; i++)  array[i] = array[i + 1];  Array.Resize(ref array, array.Length - 1);  }  }  }  IEnumerable<T> for\_each()  {  for (int i = 0; i < array.Length; i++)  yield return array[i];  }  void erase(int index)  {  for (int i = index; i < array.Length - 1; i++)  array[i] = array[i + 1];  Array.Resize(ref array, array.Length - 1);  }  public IEnumerator<T> GetEnumerator()  {  return for\_each().GetEnumerator();  }  IEnumerator IEnumerable.GetEnumerator()  {  return GetEnumerator();  }  public void IntersectWith(IEnumerable<T> other)  {  for (int i = array.Length - 1; i >= 0; i--)  if (!other.Contains(array[i]))  erase(i);  }  /// <summary>  /// Подмножество  /// </summary>  /// <param name="other"></param>  /// <returns></returns>  public bool IsProperSubsetOf(IEnumerable<T> other)  {  if (array.Length == 0 || other.ToArray().Length == 0)  return true;  for (int i = 0; i < array.Length; i++)  if (other.Contains(array[i]))  continue;  else  return false;  T[] a = other.ToArray();  for (int i = 0; i < a.Length; i++)  if (Contains(a[i]))  continue;  else  return true;  return false;  }  /// <summary>  /// Надмножество  /// </summary>  /// <param name="other"></param>  /// <returns></returns>  public bool IsProperSupersetOf(IEnumerable<T> other)  {  int otherLength = other.ToArray().Length;  if (array.Length == 0 || otherLength == 0)  return true;  if (array.Length <= otherLength)  return false;  int exist = 0, notExist = 0;  for (int i = 0; i < array.Length; i++)  if (other.Contains(array[i]))  exist++;  else  notExist++;  return exist != 0 && notExist != 0;  }  public bool IsSubsetOf(IEnumerable<T> other)  {  if (array.Length == 0)  return true;  for (int i = 0; i < array.Length; i++)  if (!other.Contains(array[i]))  return false;  return true;  }  public bool IsSupersetOf(IEnumerable<T> other)  {  if (array.Length < other.ToArray().Length)  return false;  T[] a = other.ToArray();  for (int i = 0; i < a.Length; i++)  if (!Contains(a[i]))  return false;  return true;  }  /// <summary>  /// Истина если хотя бы один элемент общий  /// </summary>  /// <param name="other"></param>  /// <returns></returns>  public bool Overlaps(IEnumerable<T> other)  {  T[] a = other.ToArray();  if (a.Length > array.Length)  {  for (int i = 0; i < a.Length; i++)  if (Contains(a[i]))  return true;  return false;  }  else  for (int i = 0; i < array.Length; i++)  if (other.Contains(array[i]))  return true;  return false;  }  public bool Remove(T item)  {  if (Contains(item))  {  int index = 0;  while (!array[index].Equals(item))  index++;  erase(index);  return true;  }  else  return false;  }  void erase(ref T[] a, int index)  {  for (int i = index; i < a.Length - 1; i++)  a[i] = a[i + 1];  Array.Resize(ref a, a.Length - 1);  }  /// <summary>  /// Истина если во обоих множествах одинаковые элементы. Дубли и порядок не важен  /// </summary>  /// <param name="other"></param>  /// <returns></returns>  public bool SetEquals(IEnumerable<T> other)  {  T[] a = other.ToArray();  for (int i = 0; i < a.Length; i++)  if (!Contains(a[i]))  return false;  for (int i = 0; i < array.Length; i++)  if (!other.Contains(array[i]))  return false;  return true;  }  /// <summary>  /// Изменяет текущий набор, чтобы он содержал только элементы,  /// которые имеются либо в текущем наборе или в указанной  /// коллекции, но не одновременно.  /// </summary>  /// <param name="other"></param>  public void SymmetricExceptWith(IEnumerable<T> other)  {  Set<T> newSet = new Set<T>();  for (int i = 0; i < array.Length; i++)  if (!other.Contains(array[i]))  newSet.Add(array[i]);  T[] a = other.ToArray();  for (int i = 0; i < a.Length; i++)  if (!Contains(a[i]))  newSet.Add(a[i]);  array = newSet.array;  }  public void UnionWith(IEnumerable<T> other)  {  Set<T> set = new Set<T>();  for (int i = 0; i < array.Length; i++)  set.Add(array[i]);  foreach (T item in other)  set.Add(item);  array = set.array;  }  public bool IsReadOnly  {  get  {  return false;  }  }  public int Count  {  get  {  return array.Length;  }  }  }  namespace Lab8  {  class Lab8  {  static void Main(string[] args)  {  Set<int> set = new Set<int> { 1, 2, 3 };  Console.Write(set.Contains(1) + " ");  HashSet<int> hash = new HashSet<int> { 1, 2, 3 };  Console.WriteLine(hash.Contains(1));  Console.WriteLine(set.Count + " " + hash.Count);  set.Add(5);  hash.Add(5);  set.ExceptWith(new int[] { 2, 3 });  hash.ExceptWith(new int[] { 2, 3 });  Console.Write(string.Join(" ", set.ToArray()) + " | " + string.Join(" ", hash.ToArray()));  set.IntersectWith(new int[] { 1, 2 });  hash.IntersectWith(new int[] { 1, 2 });  Console.WriteLine();  Console.Write(string.Join(" ", set.ToArray()) + " | " + string.Join(" ", hash.ToArray()));  Random rnd = new Random();  for (int i = 0; i < 5; i++)  {  int val = rnd.Next(0, 10);  set.Add(val);  hash.Add(val);  }  Console.WriteLine();  Console.WriteLine(new Set<int>().IsProperSubsetOf(new int[] { 1 }) + " " +  new HashSet<int>().IsProperSubsetOf(new int[] { 1 }));  Console.WriteLine(new Set<int> { 1 }.IsProperSubsetOf(new int[] { 1 }) + " " +  new HashSet<int> { 1 }.IsProperSubsetOf(new int[] { 1 }));  Console.WriteLine(new Set<int> { 1 }.IsProperSubsetOf(new int[] { 1, 2 }) + " " +  new HashSet<int> { 1 }.IsProperSubsetOf(new int[] { 1, 2 }));  Console.WriteLine(new Set<int> { 1, 2, 3 }.IsProperSubsetOf(new int[] { 1, 2, 3, 4, 5 }) + " " +  new HashSet<int> { 1, 2, 3 }.IsProperSubsetOf(new int[] { 1, 2, 3, 4, 5 }));  Console.WriteLine(new Set<int> { 1, 2, 3 }.IsProperSubsetOf(new int[] { 1, 2, 4, 5 }) + " " +  new HashSet<int> { 1, 2, 3 }.IsProperSubsetOf(new int[] { 1, 2, 4, 5 }));  Console.WriteLine(new Set<int> { 1, 2, 3 }.IsProperSupersetOf(new int[] { 1, 2, 3, 4, 5 }) + " " +  new HashSet<int> { 1, 2, 3 }.IsProperSupersetOf(new int[] { 1, 2, 3, 4, 5 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.IsProperSupersetOf(new int[] { 1, 2, 3 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.IsProperSupersetOf(new int[] { 1, 2, 3 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.IsSubsetOf(new int[] { 1, 2, 3 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.IsSubsetOf(new int[] { 1, 2, 3 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.IsSubsetOf(new int[] { 1, 2, 3, 8, 9 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.IsSubsetOf(new int[] { 1, 2, 3, 8, 9 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.IsSupersetOf(new int[] { 1, 2, 3, 8, 9 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.IsSupersetOf(new int[] { 1, 2, 3, 8, 9 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.IsSupersetOf(new int[] { 1, 2, 3 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.IsSupersetOf(new int[] { 1, 2, 3 }));  Console.WriteLine(new Set<int> { 1, 2, 3, 8 }.Overlaps(new int[] { 1, 2, 3 }) + " " +  new HashSet<int> { 1, 2, 3, 8 }.Overlaps(new int[] { 1, 2, 3 }));  Console.WriteLine(new Set<int> { 0 }.Overlaps(new int[] { 1, 2, 3 }) + " " +  new HashSet<int> { 0 }.Overlaps(new int[] { 1, 2, 3 }));  Console.WriteLine(new Set<int> { 1, 2, 3 }.SetEquals(new int[] { 2, 1, 3, 1 }) + " " +  new HashSet<int> { 1, 2, 3 }.SetEquals(new int[] { 2, 1, 3, 1 }));  set.SymmetricExceptWith(new int[] { 1, 3 });  hash.SymmetricExceptWith(new int[] { 1, 3 });  Console.WriteLine(string.Join(" ", set) + " | " + string.Join(" ", hash));  set.UnionWith(new int[] { 1, 2, 3, 4 });  hash.UnionWith(new int[] { 1, 2, 3, 4 });  Console.WriteLine(string.Join(" ", set) + " | " + string.Join(" ", hash));  Console.ReadKey(true);  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 8.1.

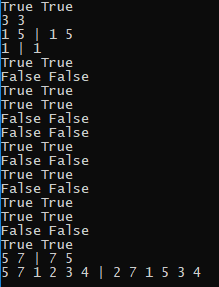


Рисунок 8.1 - Результат работы программы по классам и операциям

# 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

*Создать абстрактный класс Vehicle (транспортные средство). На его основе реализовать классы Plane (самолет), Car (автомобиль) и Ship (корабль). Классы должны иметь возможность задавать и получить координаты и параметры средств передвижения (цена, скорость, год выпуска и т. п.) с помощью свойств. Для самолета должна быть определена высота, для самолета и корабля - количество пассажиров, для корабля - порт приписки. Динамические характеристики задать с помощью методов.*

## Текст программы

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

Листинг 8. – Текст файла prog9.cs

|  |
| --- |
| using System;  namespace Lab9  {  class Lab9  {  static void Main()  {  Console.WriteLine("\*\*\*\*Самолёт\*\*\*\*");  Console.WriteLine("Характеристики самолёта");  var plane = new Plane(200) { Year = 2001/\*год производства\*/, Price = 1000000, /\*цена\*/ Height = 2000, /\*высота\*/ Passengers = 10,/\*количество пассажиров\*/ Speed = 400 /\*скорость\*/};  Console.WriteLine("Год выпуска: {0}", plane.Year.ToString());  Console.WriteLine("Стоимость самолёта {0}:", plane.Price.ToString());  Console.WriteLine("Высота полёта {0}:", plane.Height.ToString());  Console.WriteLine("Число пассажиров {0}:", plane.Passengers.ToString());  Console.WriteLine("Скорость {0}:", plane.Speed.ToString());  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.WriteLine("\*\*\*\*Машина\*\*\*\*");  Console.WriteLine("Характеристики Машина");  var car = new Car(60) { Price = 50000,/\*цена\*/ Year = 1996, /\*год производства\*/ Speed = 250 /\*скорость\*/};  car.Coordinate(40000, 30000);  Console.WriteLine("Стоимость машины {0}:", car.Price.ToString());  Console.WriteLine("Координаты машины X:{0},Y: {1}", car.CoordinateX.ToString(), car.CoordinateY.ToString());  Console.WriteLine("Год выпуска: {0}", car.Year.ToString());  Console.WriteLine("Стоимость машины {0}:", car.Price.ToString());  Console.WriteLine("Скорость {0}:", car.Speed.ToString());  var ship = new Ship(20) { Price = 5000000/\*цена\*/, Year = 1986,/\*год производства\*/ Speed = 95,/\*скорость\*/ Port = "Порт Артур"/\*порт приписки\*/ };  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.WriteLine("\*\*\*\*Корабль\*\*\*\*");  Console.WriteLine("Характеристики корабля");  Console.WriteLine("Стоимость корабля {0}:", ship.Price.ToString());  Console.WriteLine("Координаты машины X:{0},Y: {1}", ship.CoordinateX.ToString(), car.CoordinateY.ToString());  Console.WriteLine("Год выпуска: {0}", ship.Year.ToString());  Console.WriteLine("Стоимость машины {0}:", ship.Price.ToString());  Console.WriteLine("Скорость {0}:", ship.Speed.ToString());  Console.WriteLine("Порт приписки {0}:", ship.Port.ToString());  Console.Read();  }  }  //класс транспорт  internal class Vehicle  {  private double coordinateX, coordinateY;  private double price, speed, year\_of\_construction;  public double CoordinateX //координты X  {  get { return coordinateX; }  set { coordinateX = value; }  }  public double CoordinateY //координаты Y  {  get { return coordinateY; }  set { coordinateY = value; }  }  public double Price //цена  {  get { return price; }  set { price = value; }  }  public double Speed //скорость  {  get { return speed; }  set { speed = value; }  }  public double Year //дата производства  {  get { return year\_of\_construction; }  set { year\_of\_construction = value; }  }  //Метод для динамически меняющихся свойств  public void Coordinate(double CoordinateX, double CoordinateY)  {  this.CoordinateX = CoordinateX;  this.coordinateY = CoordinateY;  }  //Метод для динамически меняющихся свойств  /\* public void SpeedMethod(double Speed)  {  this.Speed = Speed;  }\*/  //конструктор  public Vehicle(double Speed)  {  this.Speed = Speed;  }  }  class Plane : Vehicle //самолет  {  public double Height { get; set; } //высота  public double Passengers { get; set; } //каличество пассажиров  //наследование конструктора из Vehicle  public Plane(double Speed)  : base(Speed)  {  Height = 300;  Passengers = 100;  }  }  class Ship : Vehicle //корабль  {  public double Passengers { get; set; } //количество пассажиров  public string Port { get; set; } //порт приписки  //наследование конструктора из Vehicle  public Ship(double Speed)  : base(Speed)  {  Passengers = 500;  Port = "порт 1";  }  }  class Car : Vehicle //машина  {  //наследование конструктора из Vehicle  public Car(double Speed)  : base(Speed)  {  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 9.1.

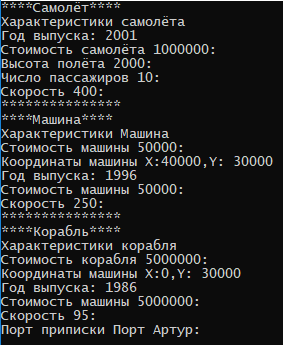


Рисунок 9.1 - Результат работы программы

# 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 Note, содержащую следующие поля:*

*- фамилия и имя;*

*- номер телефона;*

*- дата рождения (массив из трёх чисел).*

*Написать программу, которая выполняет следующие действия:*

*- ввод с клавиатуры данных в массив, состоящий из восьми элементов типа Note (записи должны быть упорядочены по дате рождения);*

*- вывод на экран информации о человеке, номер телефона которого введена с клавиатуры (если такого 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;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace Lab10  {  class Lab10  {  struct Node  {  public string name;  public string phoneNumber;  public int[] dateOfBirth;  public override string ToString()  {  return name + " " + phoneNumber + " " + dateOfBirth[0] + "." + dateOfBirth[1] + "." + dateOfBirth[2];  }  public static int CompareBirthdays(Node first, Node second)  {  if (first.dateOfBirth[2] == second.dateOfBirth[2])  {  if (first.dateOfBirth[1] == second.dateOfBirth[1])  {  return first.dateOfBirth[0].CompareTo(second.dateOfBirth[0]);  }  else return first.dateOfBirth[1].CompareTo(second.dateOfBirth[1]);  }  else return first.dateOfBirth[2].CompareTo(second.dateOfBirth[2]);  }  }  static void Main(string[] args)  {  Console.Write("Введите количество записей: ");  int count = Convert.ToInt32(Console.ReadLine());  List<Node> book = new List<Node>(count);  for (int i = 0; i < count; i++)  {  Console.WriteLine("Запись номер {0}: ", i+1);  Node node = new Node();  Console.Write("Введите имя: ");  node.name = Console.ReadLine();  Console.Write("Введите номер телефона: ");  node.phoneNumber = Console.ReadLine();  Console.Write("Введите дату рождения (ДД.ММ.ГГ): ");  string[] date = Console.ReadLine().Split('.');  node.dateOfBirth = new int[3];  node.dateOfBirth[0] = Convert.ToInt32(date[0]);  node.dateOfBirth[1] = Convert.ToInt32(date[1]);  node.dateOfBirth[2] = Convert.ToInt32(date[2]);  book.Add(node);  }  book.Sort(Node.CompareBirthdays);  Console.WriteLine("Содержимое книги:");  foreach (Node item in book)  Console.WriteLine(item.ToString());  Console.Write("Введите номер человека, которого хотите найти: ");  string searchNumber = Console.ReadLine();  foreach (Node item in book)  {  if(item.phoneNumber == searchNumber)  Console.WriteLine(item.ToString());  }    Console.Read();  }  }  } |

## Тестирование программы

Результат работы программы приведен на рисунке 10.1.

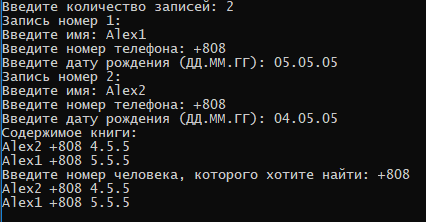


Рисунок 10.1 - Результат работы программы по структурам

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»).