Лабораторная работа №1

Условие:

1. Запрашивает с клавиатуры три вещественных числа и выводит следующее сообщение (*a+(b+c))=(a+c+b)* (вещественные числа выводятся с точностью до 4 знаков после запятой);
2. Дано трехзначное число. Найти число, полученное при прочтении его цифр справа налево;
3. Периметр треугольника, заданного координатами вершин x1, y1, x2, y2, x3, y3;
4. Определить силу притяжения *F* между телами массы *т*1 и *т*2, находящимися на расстоянии *r* друг от друга;
5. Пусть даны четыре целых числа (hour, min, sec, time). Первые три из них (hour, min, sec) - это время запуска ракеты в часах, минутах и секундах, четвертое (time) определяет время полета в секундах. Найдите и напечатайте время возвращения ракеты на землю;
6. ;
7. *а*64 за шесть операции, *а* – действительное число, при этом, не пользуясь никакими другими арифметическими операциями, кроме умножения.

Решение:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp7

{

class Program

{

static void Main(string[] args)

{

//Задание 1

Console.WriteLine("Задание 1:");

double a, b, c;

Console.WriteLine("Введите a, b и c: ");

Console.WriteLine("a = {0}", a = Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("b = {0}", b = Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("c = {0}", c = Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Полученное выражение: ");

Console.WriteLine("({0:0.0000}+({1:0.0000}+{2:0.0000}))=({0:0.0000}+{2:0.0000}+{1:0.0000})", a, b, c);

//Задание 2

Console.WriteLine("Задание 2:");

Console.WriteLine("Введите трёхзначное число: ");

string three = Console.ReadLine();

char[] arr = three.ToCharArray();

Array.Reverse(arr);

Console.WriteLine(arr);

//Задание 3

Console.WriteLine("Задание 3:");

double x1, x2, x3, y1, y2, y3;

Console.WriteLine("Задайте координаты треугольника: x1, y1, x2, y2, x3, y3");

Console.WriteLine("x1 = {0}", x1 = double.Parse(Console.ReadLine()));

Console.WriteLine("y1 = {0}", y1 = double.Parse(Console.ReadLine()));

Console.WriteLine("x2 = {0}", x2 = double.Parse(Console.ReadLine()));

Console.WriteLine("y2 = {0}", y2 = double.Parse(Console.ReadLine()));

Console.WriteLine("x3 = {0}", x3 = double.Parse(Console.ReadLine()));

Console.WriteLine("y3 = {0}", y3 = double.Parse(Console.ReadLine()));

double x1y1x2y2 = Math.Sqrt(Math.Pow((x2 - x1), 2) + (Math.Pow((y2 - y1), 2)));

double x2y2x3y3 = Math.Sqrt(Math.Pow((x3 - x2), 2) + (Math.Pow((y3 - y2), 2)));

double x3y3x1y1 = Math.Sqrt(Math.Pow((x3 - x1), 2) + (Math.Pow((y3 - y1), 2)));

double Px1y1x2y2x3y3 = x1y1x2y2 + x2y2x3y3 + x3y3x1y1;

Console.WriteLine("Периметр треугольника = {0}", Px1y1x2y2x3y3);

//Задание 4

Console.WriteLine("Задание 4:");

double G = (Math.Pow(6.7385, (10 ^ -11))), F, m1, m2, r;

Console.WriteLine("Задайте вес тел и расстояние их друг от друга: m1, m2, r");

Console.WriteLine("m1 = {0}", m1 = double.Parse(Console.ReadLine()));

Console.WriteLine("m2 = {0}", m2 = double.Parse(Console.ReadLine()));

Console.WriteLine("r = {0}", r = double.Parse(Console.ReadLine()));

Console.WriteLine("Сила притяжения = {0} кг\*м/с^2", F = G \* ((m1 \* m2)/(Math.Pow(r, 2))));

//Задание 5

Console.WriteLine("Задание 5:");

int hour, min, sec, time, time1, hour1, min1, sec1;

Console.WriteLine("Введите часы, минуты и секунды запуска ракеты: hour, min, sec. После - время полета в секундах: time");

Console.WriteLine("hour = {0}", hour = Int32.Parse(Console.ReadLine()));

Console.WriteLine("min = {0}", min = Int32.Parse(Console.ReadLine()));

Console.WriteLine("sec = {0}", sec = Int32.Parse(Console.ReadLine()));

Console.WriteLine("time = {0}", time = Int32.Parse(Console.ReadLine()));

time1 = hour \* 360 + min \* 60 + sec + time;

Console.WriteLine("Общее время полета в секундах = {0}", time1);

hour1 = time1 / 360;

min1 = (time1 % 360) / 60;

sec1 = (time1 - hour \* 360 - min \* 60) % 60;

Console.WriteLine("Время прилета {0:00}:{1:00}:{2:00}", hour1, min1, sec1);

//Задание 6

Console.WriteLine("Задание 6:");

double e, x, y;

e = 2.7;

x = 6.4;

y = (Math.Pow(e, x)) / (Math.Cos(Math.Sqrt(x - 1))) + (2 \* Math.Atan(Math.Pow(x, 2))) / (1 - x);

Console.WriteLine("Значение функции y = {0}", y);

//Задание 7

Console.WriteLine("Задание 7:");

decimal a7;

Console.WriteLine("Введите а: ");

Console.WriteLine("a = {0}", a7 = decimal.Parse(Console.ReadLine()));

a7 = a7 \* a7; //2 степень

a7 = a7 \* a7; //4 степень

a7 = a7 \* a7; //8 степень

a7 = a7 \* a7; //16 степень

a7 = a7 \* a7; //32 степень

a7 = a7 \* a7; //64 степень

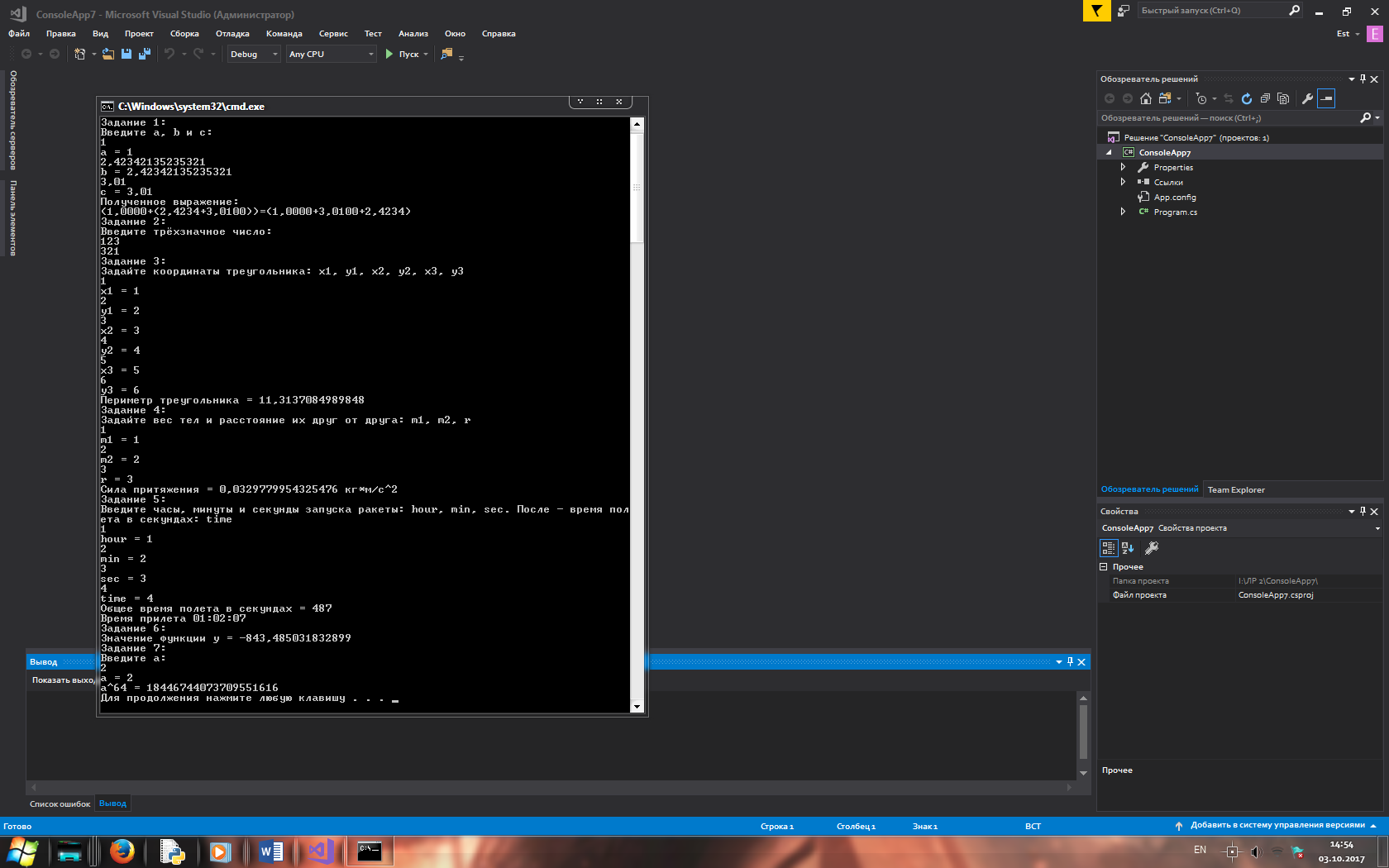
Console.WriteLine("a^64 = {0}", a7);

}

}

}

Тестирование:



Контрольные вопросы:

1. Прописные и строчные латинские буквы и буквы национальных алфавитов (включая кириллицу);
   * + 1. арабские цифры от 0 до 9, шестнадцатеричные цифры от A до F;
       2. специальные знаки: " { } , | ; [ ] ( ) + - / % \* . \ ' : ? < = > ! & ~ ^ @ \_
       3. пробельные символы: пробел, символ табуляции, символ перехода на новую строку.
2. В С# типы делятся на две группы: базовые типы, предлагаемые языком, и типы, определяемые пользователем. Также типы С# разбиваются на две другие категории: размерные типы и ссылочные типы. Почти все базовые типы являются размерными типами. Исключение составляют типы Object и String. Принципиальное различие между размерными и ссылочными типами состоит в способе хранения их значений в памяти. В первом случае фактическое значение хранится в стеке. Адрес переменной ссылочного типа тоже хранится в стеке, но сам объект хранится в куче.
3. Преобразование типов данных осуществляется с помощью класса Convert или метода Parse. Для преобразования строкового представления целого числа в тип int используется метод int.Parse(), который реализован для всех числовых типов данных. Если требуется преобразовать строковое представление в вещественное, то используется метод float.Parse() или double.Parse(). В случае, если соответствующее преобразование выполнить невозможно, то выполнение программы прерывается и генерируется исключение System.FormatExeption (входная строка имела неверный формат).
4. Пробелы внутри имен не допускаются. Язык С# не налагает никаких ограничений на длину имен, однако для удобства чтения и записи кода не стоит делать их слишком длинными. Для улучшения читабельности кода программным элементам следует давать осмысленные имена, составленные в соответствии с определенными правилами. Существует несколько видов нотаций – соглашений о правилах создания имен.
5. Необходимость использования комментариев в программе – для быстрой дезактивации куска кода, без удаления его из программы.
6. Назначение методов Write и WriteLine – для вывода данных.
7. Console.writeline и write
8. If, swich
9. Math.cos, Math.Pow, Math.ABS, Math.sqrt, Math.sqr, Math.sin
10. WriteLine
11. Границы лексем определяются другими лексемами, такими, как разделители или знаки операций. В свою очередь лексемы входят в состав выражений (выражение задает правило вычисления некоторого значения) и операторов (оператор задает законченное описание некоторого действия).
12. Пример преобразования – int.Parse().
13. ++ и --. Или переменная++, --