Федеральное государственное бюджетное образовательное учреждение высшего образования «Пермский национальный исследовательский политехнический университет»

Лабораторная работа №1

По дисциплине

“Объектно-ориентированное программирование”

Тема

“Выполнение программы простой структуры. Вычисление выражений с использованием стандартных функций”

Вариант 12

Выполнил работу

студент группы РИС-19-1б

Мазитов Т.Э.

Проверила

доцент кафедры ИТАС

Викентьева О.Л.

Работу выполнил:

Пермь 2020

**Постановка задачи**

1. Для задачи 1 определить тип заданных выражений и найти их значения.
2. Составить систему тестов и вычислить полученное выражение для нескольких значений Х, определить при каких Х выражение не может быть вычислено.
3. Для задачи 2 записать выражение, зависящее от координат точки X1 и Y1 и принимающее значение TRUE, если точка принадлежит заштрихованной области, и FALSE, если не принадлежит.
4. Составить систему тестов и вычислить полученное выражение для нескольких точек, принадлежащих и не принадлежащих заштрихованной области.
5. Для задачи 3 вычислить значение выражения, используя различные вещественные типы данных (float и double).
6. Результаты всех вычислений вывести на печать.
7. Объяснить полученные результаты.

|  |  |  |
| --- | --- | --- |
| Задача 1 | Задача 2 | Задача 3 |
| 1) n++\*m  2) m--<n  3)++m>n  4) |  | а=10, b=0.01 |

**Формулы, используемые при решении**

* **–** формула круга; определяет множество точек, находящихся внутри круга или на его границе;

**Используемые методы класса Math**

* Pow(n, k) – возводит число n в степень k;
* Round(n, k) – округляет число n до k знаков после запятой;
* Sqrt(n) – возвращает корень числа n;
* Abs(n) – возвращает модуль числа n.

**Код программ**

**Задача 1**

using System;

using System.Collections.Generic;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

Task();

}

static void Read(ref float m, ref float n)

{

bool flag;

do

{

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

string buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag = float.TryParse(buff, out m);

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

buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag &= float.TryParse(buff, out n);

} while (!flag);

}

static void Read\_x(ref double x)

{

bool flag;

do

{

Console.WriteLine("Введите число x [-1;INF): ");

string buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag = double.TryParse(buff, out x);

} while (x < -1 || !flag);

}

static void TestTask4()

{

int count = 0;

var testDict = new Dictionary<double, double>();

testDict[-16] = -1;

testDict[-5] = -1;

testDict[-2] = -1;

testDict[-1.1] = -1;

testDict[-1] = 1;

testDict[-0.9] = 1.172;

testDict[0] = 0;

testDict[0.9] = 2.269;

testDict[1] = 2.414;

testDict[1.1] = 2.557;

testDict[2] = 3.786;

testDict[5] = 7.549;

testDict[16] = 20.243;

Console.WriteLine("\nСписок тестов");

Console.WriteLine("{0,7}{1,15}", "X", "Результат");

foreach (var i in testDict)

{

Console.WriteLine("{0,7}{1,15}", i.Key, i.Value);

}

//List<double> testSet = new List<double> { -5, -2, -1.1, -1, -0.9, 0, 0.9, 1, 1.1, 2, 5 };

Console.WriteLine();

Console.WriteLine("ТЕСТИРОВАНИЕ".PadLeft(23,'-').PadRight(35,'-'));

Console.WriteLine("\tX\tER\t AR");

foreach (var i in testDict)

{

var x = i.Key;

var res = Task4(ref x);

res = Math.Round(res, 3);

if ( res == i.Value) count++;

Console.WriteLine("{0,9}{1,9}{2,11}", i.Key, i.Value, res);

}

Console.WriteLine("\n\nРезультаты тестирования:\n\tВсего тестов:\t{0}\n\tУспешно:\t{1}\n\tПровалено:\t{2}", testDict.Count, count, (testDict.Count - count));

}

static double Task4(ref double x)

{

if (x < -1)

{

return -1;

}

else

{

var res = Math.Sqrt(x + Math.Sqrt(Math.Sqrt(Math.Abs(x)))) + Math.Abs(x);

return res;

}

}

static void Task()

{

float m = 0, n = 0;

double x = 0;

Read(ref m, ref n);

var res1 = n++ \* m;

Console.WriteLine("\t1) Результат выражения \"(n++\*m)\" - " + (res1) + ", тип выражения - " + res1.GetType());

var res2 = m-- < n;

Console.WriteLine("\t2) Результат выражения \"(m--<n)\" - " + (res2) + ", тип выражения - " + res2.GetType());

var res3 = ++m > n;

Console.WriteLine("\t3) Результат выражения \"(++m>n)\" - " + (res3) + ", тип выражения - " + res3.GetType());

Read\_x(ref x);

var res4 = Task4(ref x);

Console.WriteLine("\t4) Результат выражения - " + res4 + " , тип выражения - " + res4.GetType());

TestTask4();

}

}

}

**Задача 2**

using System;

using System.Collections.Generic;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

Task();

}

static bool Circle(double x1, double y1, double r)

{

//var c = Math.Pow(x1, 2) + Math.Pow(y1, 2);

if ((Math.Pow(x1, 2) + Math.Pow(y1, 2) <= Math.Pow(r,2)) && (x1 \* y1 <= 0))

{

return true;

}

else return false;

}

static void Read(ref double x1, ref double y1, ref double r)

{

bool flag;

do

{

Console.WriteLine("Введите число x1: ");

string buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag = double.TryParse(buff, out x1);

Console.WriteLine("Введите число y1: ");

buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag &= double.TryParse(buff, out y1);

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

buff = Console.ReadLine();

buff = buff.Replace(".", ",");

flag &= double.TryParse(buff, out r);

} while (!flag);

}

static void Test()

{

int count = 0;

var testDict = new Dictionary<List<double>, bool>();

testDict[new List<double> { 5, 5, 1 }] = false;

testDict[new List<double> { -1, -1, 1 }] = false;

testDict[new List<double> { -1, 1, 1 }] = false;

testDict[new List<double> { 1, -1, 1 }] = false;

testDict[new List<double> { 1, 1, 1 }] = false;

testDict[new List<double> { 0, 0, 1 }] = true;

testDict[new List<double> { 1, 0, 1 }] = true;

testDict[new List<double> { 0, 1, 1 }] = true;

testDict[new List<double> { -1, 0, 1 }] = true;

testDict[new List<double> { 0, -1, 1 }] = true;

testDict[new List<double> { 0.99, 0, 1 }] = true;

testDict[new List<double> { 0, 0.99, 1 }] = true;

testDict[new List<double> { -0.99, 0, 1 }] = true;

testDict[new List<double> { 0, -0.99, 1 }] = true;

testDict[new List<double> { 0.1, 0.1, 1 }] = false;

testDict[new List<double> { -0.1, -0.1, 1 }] = false;

testDict[new List<double> { 0.1, -0.1, 1 }] = true;

testDict[new List<double> { -0.1, 0.1, 1 }] = true;

testDict[new List<double> { 0.8, 0.6, 1 }] = false;

testDict[new List<double> { -0.8, 0.6, 1 }] = true;

testDict[new List<double> { 0.8, -0.6, 1 }] = true;

testDict[new List<double> { -0.8, -0.6, 1 }] = false;

Console.WriteLine("\nСписок тестов");

Console.WriteLine("{0,7}{1,7}{2,7}{3,10}","X","Y","R","Попадание");

foreach (var i in testDict)

{

Console.WriteLine("{0,7}{1,7}{2,7}{3,10}",i.Key[0], i.Key[1], i.Key[2], i.Value);

}

Console.WriteLine("\n Входные данные (X,Y,R)\t Ожидаемый результат\tРеальный результат");

foreach (var i in testDict)

{

var res = Circle(i.Key[0], i.Key[1], i.Key[2]);

if (res == i.Value) count++;

Console.WriteLine("{0,7},{1,7},{2,7}{3,21}{4,22}", i.Key[0], i.Key[1], i.Key[2],i.Value, res);

}

Console.WriteLine("\n\nРезультаты тестирования:\n\tВсего тестов:\t{0}\n\tУспешно:\t{1}\n\tПровалено:\t{2}", testDict.Count, count, (testDict.Count - count));

}

static void Task()

{

double x1 = 0, y1 = 0, r = 0;

Read(ref x1, ref y1, ref r);

if (Circle(x1, y1, r))

{

Console.WriteLine("\nТочка с координатами ({0},{1}) принадлежит области с радиусом {2}",x1,y1,r);

}

else Console.WriteLine("\nТочка с координатами ({0},{1}) НЕ принадлежит области с радиусом {2}",x1,y1,r);

Test();

}

}

}

**Задача 3**

using System;

namespace Lab1

{

class Program

{

static void Main(string[] args)

{

Task();

}

static void Equation(double a, double b)

{

double res1 = (Math.Pow((a + b), 4) - Math.Pow(a, 4)) / (6 \* Math.Pow(a, 2) \* Math.Pow(b, 2) + 4 \* a \* Math.Pow(b, 3) + Math.Pow(b, 4) + 4 \* b \* Math.Pow(a, 3));

double res2 = ((a+b)\* (a + b) \* (a + b) \* (a + b) - a\*a\*a\*a ) / (6 \* a\*a \* b\*b + 4 \* a \* b\*b\*b + b\*b\*b\*b + 4 \* b \* a\*a\*a);

Console.WriteLine("С использование Math.Pow: "+res1);

Console.WriteLine("Без использования Math.Pow: "+ res2);

}

static void Equation(float a, float b)

{

float c = (float)(Math.Pow((a + b), 4));

float d = (float)(Math.Pow(a, 4));

float e = (float)(Math.Pow(a, 2));

float f = (float)(Math.Pow(b, 2));

float g = (float)(Math.Pow(b, 3));

float h = (float)(Math.Pow(b, 4));

float i = (float)(Math.Pow(a, 3));

float res1 = (float)(c - d) / (6 \* e \* f + 4 \* a \* g + h + 4 \* b \* i);

float res2 = (float)((a + b) \* (a + b) \* (a + b) \* (a + b) - a \* a \* a \* a) / (6 \* a \* a \* b \* b + 4 \* a \* b \* b \* b + b \* b \* b \* b + 4 \* b \* a \* a \* a);

Console.WriteLine("С использование Math.Pow: " + res1);

Console.WriteLine("Без использования Math.Pow: "+ res2);

}

static void Task()

{

double a = 10d, b = 0.01d;

float c = 10f, d = 0.01f;

Console.WriteLine("\nРезультат с типом double: ");

Equation(a, b);

Console.WriteLine("\nРезультат с типом float: ");

Equation(c, d);

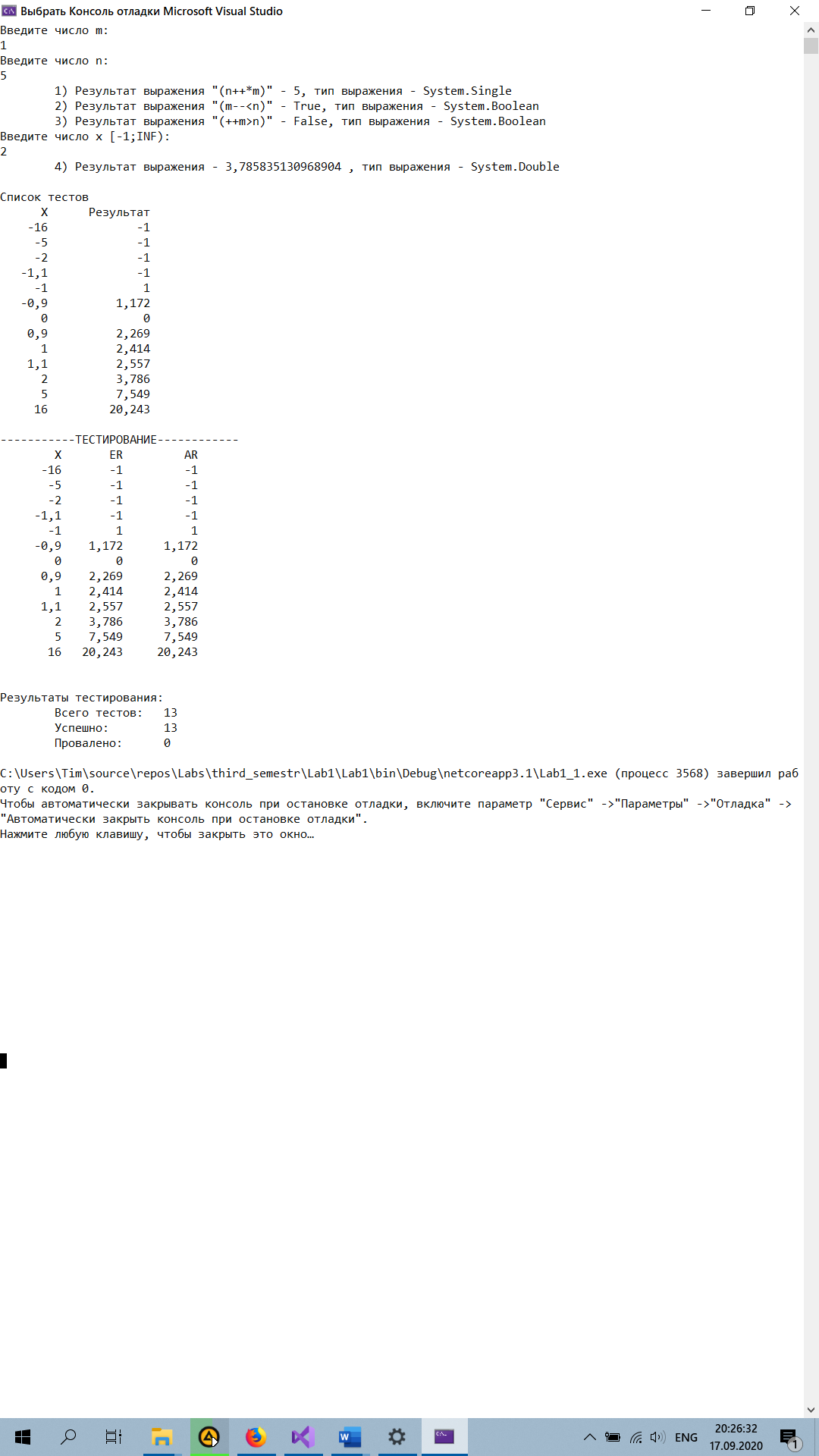
}

}

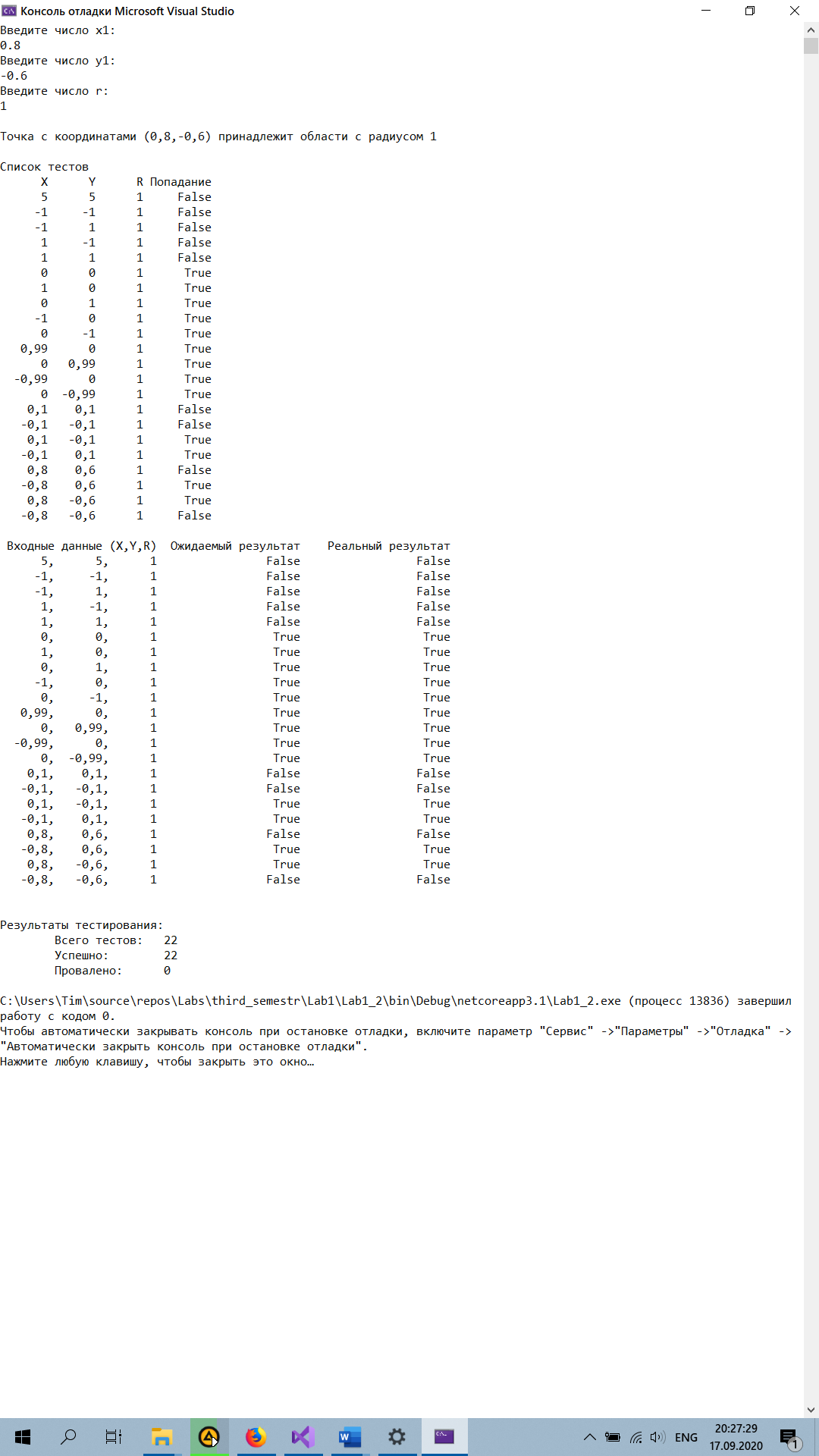
}

**Скриншот работы программы**

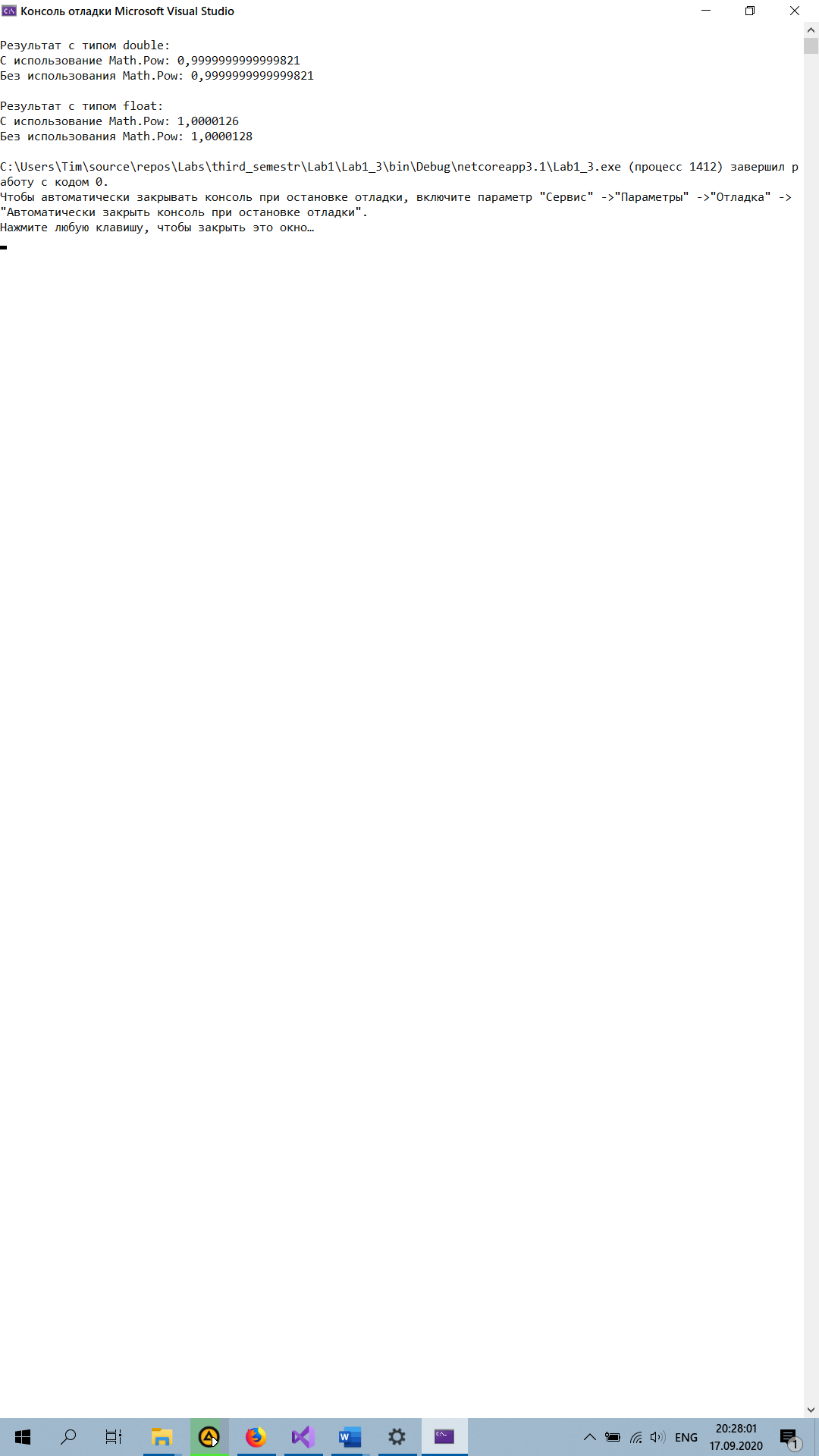
**Задача 1**

****

**Задача 2**



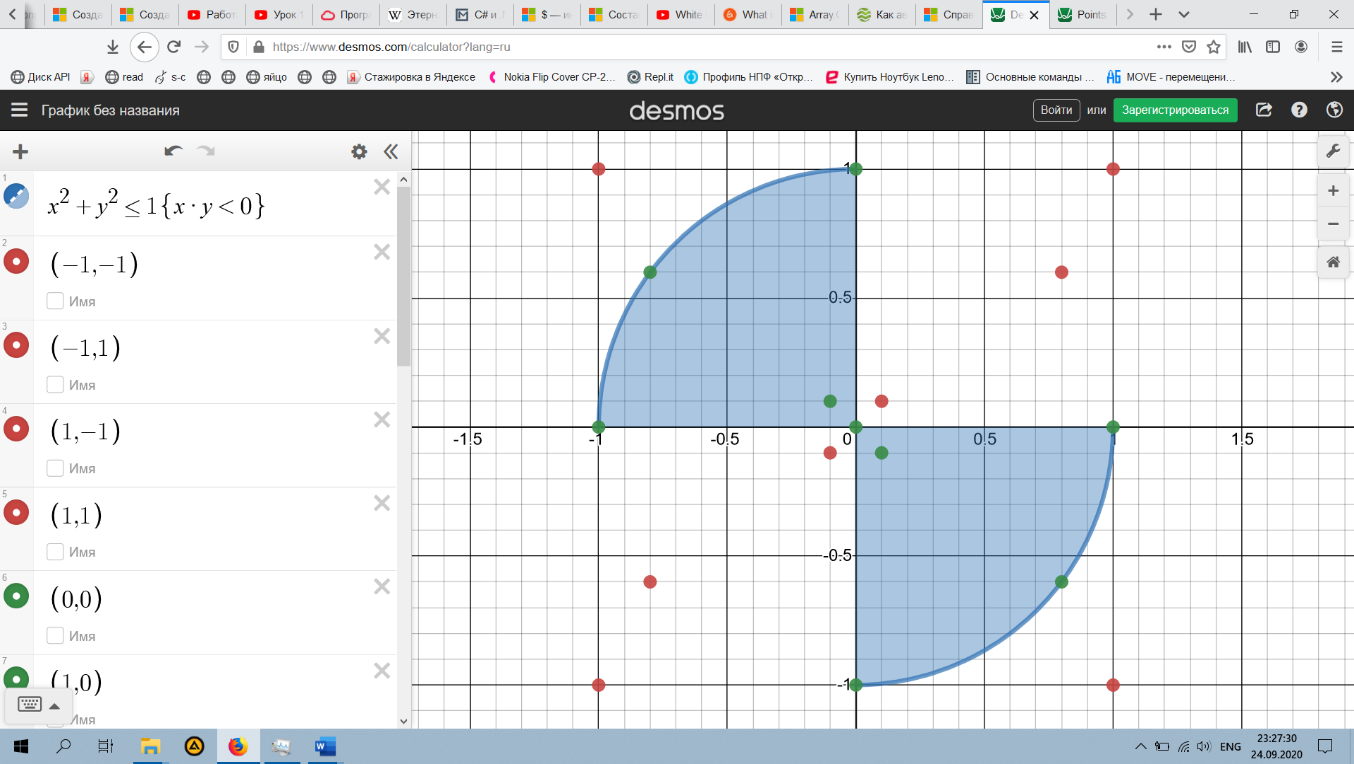
**Задача 3**



**Тестирование**

Задача 2

|  |  |  |  |
| --- | --- | --- | --- |
| Тест | Ввод | ER | AR |
| 1 | 5, 5, 1 | Точка не входит | Точка не входит |
| 2 | -1, -1, 1 | Точка не входит | Точка не входит |
| 3 | -1, 1, 1 | Точка не входит | Точка не входит |
| 4 | 1, -1, 1 | Точка не входит | Точка не входит |
| 5 | 1, 1, 1 | Точка не входит | Точка не входит |
| 6 | 0, 0, 1 | Точка входит | Точка входит |
| 7 | 1, 0, 1 | Точка входит | Точка входит |
| 8 | 0, 1, 1 | Точка входит | Точка входит |
| 9 | -1, 0, 1 | Точка входит | Точка входит |
| 10 | 0, -1, 1 | Точка входит | Точка входит |
| 11 | 0,99, 0, 1 | Точка входит | Точка входит |
| 12 | 0, 0,99, 1 | Точка входит | Точка входит |
| 13 | -0,99, 0, 1 | Точка входит | Точка входит |
| 14 | 0, -0,99, 1 | Точка входит | Точка входит |
| 15 | 0,1, 0,1, 1 | Точка не входит | Точка не входит |
| 16 | -0,1, -0,1, 1 | Точка не входит | Точка не входит |
| 17 | 0,1, -0,1, 1 | Точка входит | Точка входит |
| 18 | -0,1, 0,1, 1 | Точка входит | Точка входит |
| 19 | 0,8, 0,6, 1 | Точка не входит | Точка не входит |
| 20 | -0,8, 0,6, 1 | Точка входит | Точка входит |
| 21 | 0,8, -0,6, 1 | Точка входит | Точка входит |
| 22 | -0,8, -0,6, 1 | Точка не входит | Точка не входит |



Проверка достаточности тестов (по критериям черного ящика)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| x<0 |  | + | + |  |  |  |  |  | + |  |  |  | + |  |  | + |  | + |  | + |  | + |
| x=0 |  |  |  |  |  | + |  | + |  | + |  | + |  | + |  |  |  |  |  |  |  |  |
| 0<x<1 |  |  |  |  |  |  |  |  |  |  | + |  |  |  | + |  | + |  | + |  | + |  |
| x=1 |  |  |  | + | + |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| x>1 | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| y<0 |  | + |  | + |  |  |  |  |  | + |  |  |  | + |  | + | + |  |  |  | + | + |
| y=0 |  |  |  |  |  | + | + |  | + |  | + |  | + |  |  |  |  |  |  |  |  |  |
| 0<y<1 |  |  |  |  |  |  |  |  |  |  |  | + |  |  | + |  |  | + | + | + |  |  |
| y=1 |  |  | + |  | + |  |  | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| y>1 | + |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| r=1 | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + | + |
| Обл. | - | - | - | - | - | + | + | + | + | + | + | + | + | + | - | - | + | + | - | + | + | - |

Проверка достаточности тестов (по критериям белого ящика)

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | Т1-T5, T15, T16, T19, T22 | Т6-T14, T17, T18, T20, T21 |
| if (((Math.Pow(x1, 2) + Math.Pow(y1, 2) <= Math.Pow(r, 2))) && (x1 \* y1 <= 0)) | + |  | + |
| - | + |  |