**Оглавление**

[1. Вычисление значений выражений. 3](#_Toc179328260)

[1.1.Задание 3](#_Toc179328261)

[1.2.Код программы 3](#_Toc179328262)

[1.3.Тестирование 4](#_Toc179328263)

[2. Вычисление значения функции, заданной её графиком 5](#_Toc179328264)

[2.1.Задание 5](#_Toc179328265)

[2.2.Код программы 5](#_Toc179328266)

[2.3.Тестирование 6](#_Toc179328267)

[3. Проверка принадлежности точки заданной области на плоскости 8](#_Toc179328268)

[3.1.Задание 8](#_Toc179328269)

[3.2.Код программы 9](#_Toc179328270)

[3.3.Тестирование 10](#_Toc179328271)

[4. Вывод таблицы значений функции, заданной её графиком 11](#_Toc179328272)

[4.1.Задание 11](#_Toc179328273)

[4.2.Код приложения 11](#_Toc179328274)

[4.3.Тестирование 13](#_Toc179328275)

[5. Проверка серии «выстрелов по мишени» 13](#_Toc179328276)

[5.1.Задание 13](#_Toc179328277)

[5.2.Код программы 14](#_Toc179328278)

[5.3.Тестирование 16](#_Toc179328279)

[6. Вычисление суммы ряда с заданной точностью 16](#_Toc179328280)

[6.1.Задание 16](#_Toc179328281)

[6.2.Код программы 17](#_Toc179328282)

[6.3.Тестирование 18](#_Toc179328283)

[7. Вывод таблицы значений функции, вычисляемой с помощью ряда Тейлора 19](#_Toc179328284)

[7.1. Задание 19](#_Toc179328285)

[7.2.Код приложения 20](#_Toc179328286)

[7.3.Тестирование 21](#_Toc179328287)

# 

# 1. Вычисление значений выражений.

1.1.Задание:Ра Разработайте приложение для вычисления значений и по формулам: , .

***Исходные данные:***

а – альфа, переменная, необходимая для вычисления значений и

***Результаты:***

– значение по формуле  с заданным α;

– значение по формуле  с заданным α;

## 1.2.Код программы:

***Листинг 1 – Код программы Lab01\_01***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

double z1, z2, a;

Console.WriteLine("Добро пожаловать в программу решения значений Z1 и Z2 по формулам:\nZ1 = (Cos(a) + Sin(a))/(Cos(a) - Sin(a))\nZ2 = Tan(2a) + Sec(2\*a)\nВведите значение а:");

a = Convert.ToDouble(Console.ReadLine());

z1 = (Math.Cos(a) + Math.Sin(a))/(Math.Cos(a) - Math.Sin(a)) ;

Console.WriteLine("Z1 = " + z1);

z2 = Math.Tan(2\*a) + (1/Math.Cos(2\*a));

Console.WriteLine("Z2 = " + z2);

}

}

}

## 1.3.Тестирование:

Контрольный пример 1:

Исходные данные:

a = 10

Результат работы программы для указанных исходных данных приведён на рисунке 1.1

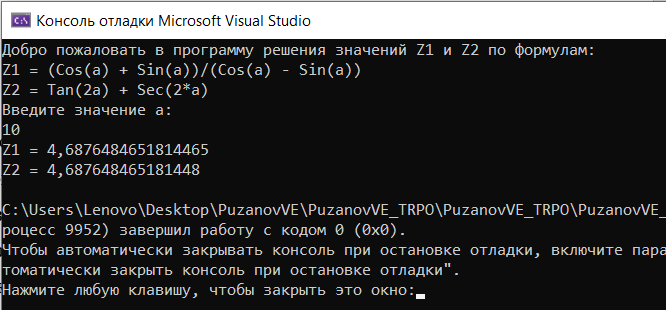


Рисунок 1.1 – Результат работы приложения Lab01\_01

Контрольный пример 2:

Исходные данные:

a = 1;

Результат работы программы для указанных исходных данных приведён на рисунке 1.2

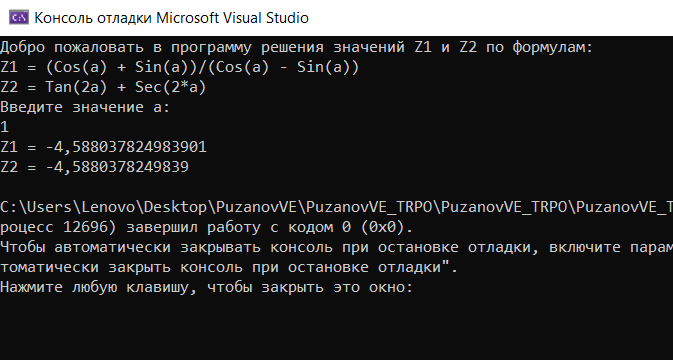
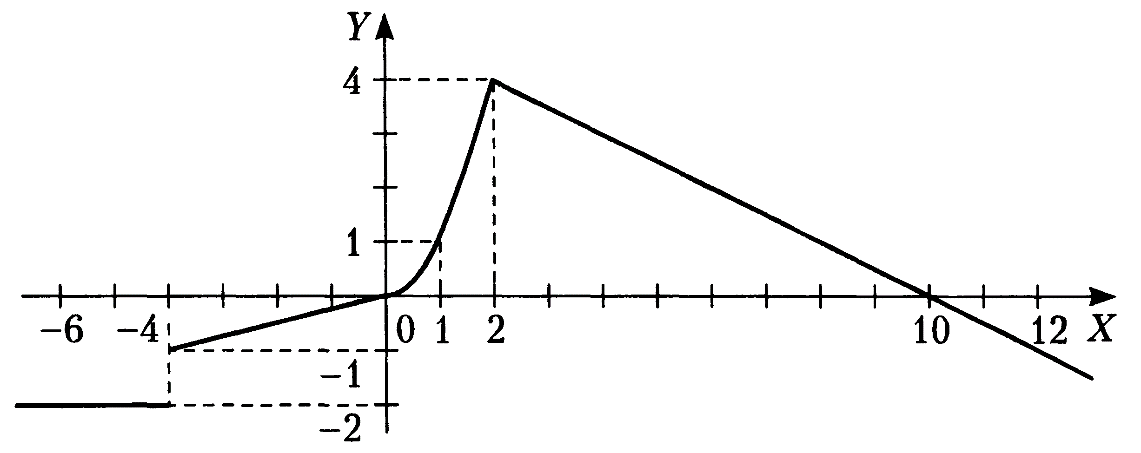


Рисунок 1.2 – Результат работы приложения Lab01\_01

# 2. Вычисление значения функции, заданной её графиком

2.1.Задание:Разработайте приложение для вычисления значения функции  в зависимости от одного заданного пользователем значения аргумента . Другие дополнительные данные, необходимые для вычисления значения функции, также вводятся пользователем.

Функция  задана графическим способом.



***Исходные данные:***

x – координата, в зависимости от которого вычисляется значение функции 

***Результаты:***

– значение функции в зависимости от заданного х;

## 2.2.Код программы:

***Листинг 2 – Код программы Lab01\_02***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

Console.WriteLine("Добро пожаловать в приложение для вычисления значения функции y = F(x)\nВведите х:");

double x = Convert.ToDouble(Console.ReadLine());

double y;

if (x <= -4)

{

y = -2;

Console.WriteLine("F(x) = -2");

Console.WriteLine("y = " + y);

}

else if (x <= 0)

{

y = x / 4;

Console.WriteLine("F(x) = x / 4");

Console.WriteLine("y = " + y);

}

else if (x <= 2)

{

y = Math.Pow(x, 2);

Console.WriteLine("F(x) = x^2");

Console.WriteLine("y = " + y);

}

else

{

y = (-x + 10) / 2;

Console.WriteLine("F(x) = (-x + 10) / 2");

Console.WriteLine("y = " + y);

}

}

}

}

## 2.3.Тестирование:

Контрольный пример 1:

Исходные данные:

x = -10

Результат работы программы для указанных исходных данных приведён на рисунке 2.1

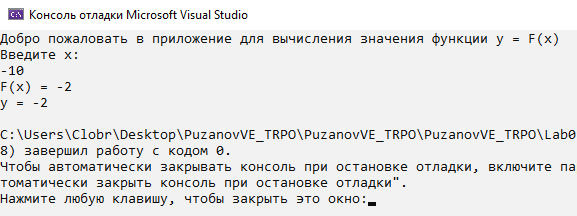


Рисунок 2.1 – Результат работы приложения Lab01\_02

Контрольный пример 2:

Исходные данные:

x = -2

Результат работы программы для указанных исходных данных приведён на рисунке 2.2

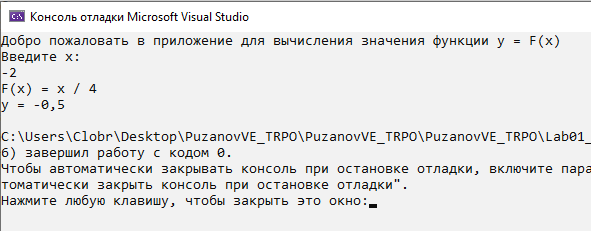


Рисунок 2.2 – Результат работы приложения Lab01\_02

Контрольный пример 3:

Исходные данные:

x = 1

Результат работы программы для указанных исходных данных приведён на рисунке 2.3

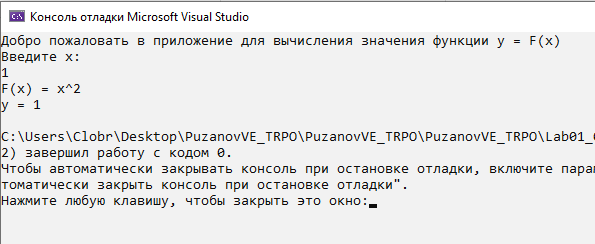


Рисунок 2.3 – Результат работы приложения Lab01\_02

Контрольный пример 4:

Исходные данные:

x = 15

Результат работы программы для указанных исходных данных приведён на рисунке 2.4

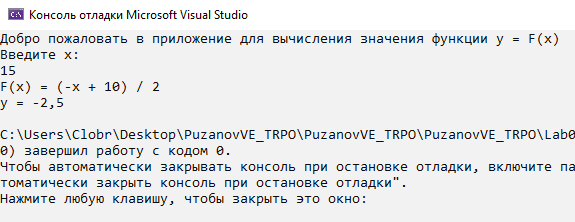
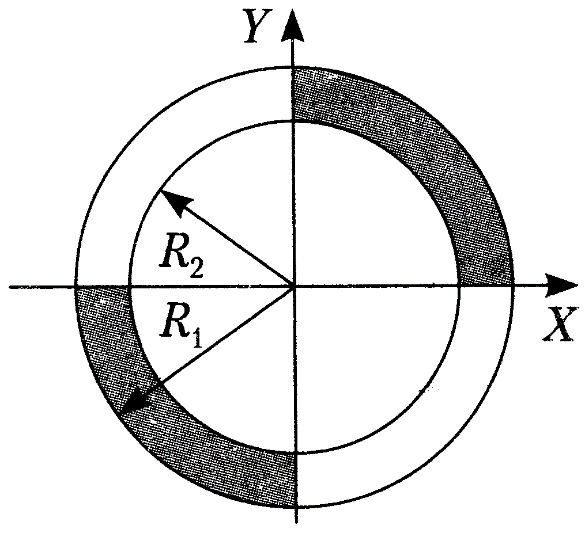


Рисунок 2.4 – Результат работы приложения Lab01\_02

# 3. Проверка принадлежности точки заданной области на плоскости

3.1.Задание:Разработайте приложение, которое позволяет проверить, принадлежит ли точка с заданными координатами  и  некоторой ограниченной области на плоскости.

Область задана графическим способом. Координаты точки и дополнительные данные, определяющие геометрию области и необходимые для решения задачи, вводятся пользователем.

Схема, на которой заданная область выделена штриховкой. ****

***Исходные данные:***

R1 – Радиус 1й (Большей) окружности;

R2 – Радиус 2й (Малой) окружности;

x – координата х;

y – координата y;

***Результаты:***

Вывод пользователю сообщение о промахе или попадании в область.

## 3.2.Код программы:

***Листинг 3 – Код программы Lab01\_03***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

double R1, R2, x, y, cord;

Console.WriteLine("Введите радиус первой (малой) окружности");

R1 = Math.Abs(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Введите радиус второй (большей) окружности");

R2 = Math.Abs(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Введите X");

x = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите Y");

y = Convert.ToDouble(Console.ReadLine());

if (((x <= 0)&&(y <= 0))||((x >= 0)&&(y >= 0)))

{

cord = x \* x + y \* y;

if (((R1 \* R1) <= cord) && (cord <= (R2\*R2)))

{

Console.WriteLine("Точка лежит в нужной области");

}

else

{

Console.WriteLine("Точка не лежит в нужной области");

}

}

else

{

Console.WriteLine("Точка не лежит в нужной области");

}

}

}

}

## 3.3.Тестирование:

Контрольный пример 1:

Исходные данные:

R1 = 2;

R2 = 4;

x = 2;

y = 2;

Результат работы программы для указанных исходных данных приведён на рисунке 3.1

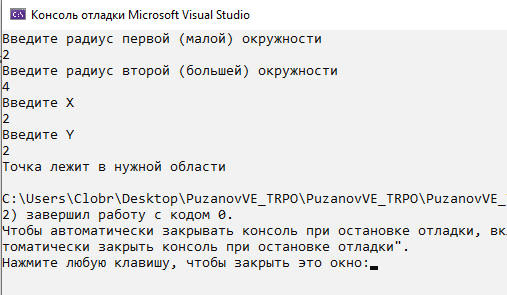


Рисунок 3.1 – Результат работы приложения Lab01\_03

Контрольный пример 2:

Исходные данные:

R1 = 2;

R2 = 4;

x = -2;

y = 2;

Результат работы программы для указанных исходных данных приведён на рисунке 3.2

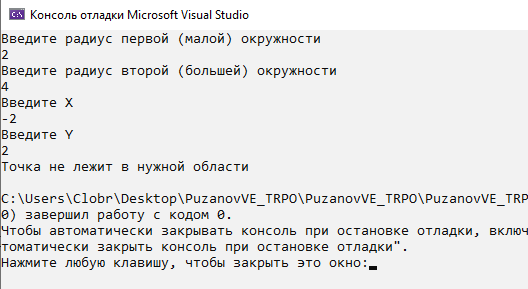
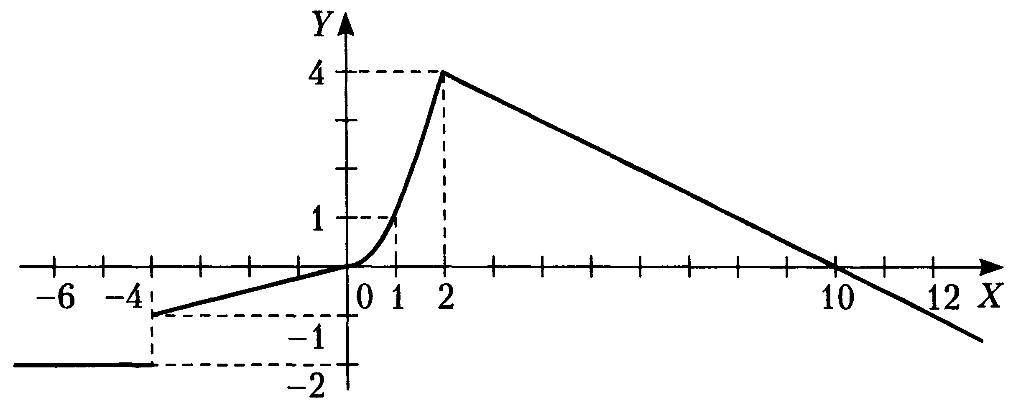


Рисунок 3.2 – Результат работы приложения Lab01\_03

# 4. Вывод таблицы значений функции, заданной её графиком

4.1.Задание***:*** Разработайте приложение для вывода таблицы значений функции  в точках, полученных делением отрезка  на  равных частей. Все данные, необходимые для вычисления значений функции и вывода таблицы, вводятся пользователем.

Функция  задана графическим способом.



***Исходные данные:***

a – начало отрезка;

b – Конец отрезка;

N – Количество частей;

***Результаты:***

Вывод таблицы значений функции  в точках, полученных делением отрезка  на  равных частей.

4.2.Код приложения*:*

***Листинг 4 – Код приложения Lab01\_04***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

Console.WriteLine("Вывод таблицы значений функции, заданной её графиком");

double y, x, a, b, AB, N, j;

Console.WriteLine("Введите начало отрезка:");

a = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите конец отрезка:");

b = Convert.ToDouble(Console.ReadLine());

AB = b - a;

if (AB <= 0)

{

Console.WriteLine("Не существующий отрезок");

}

else

{

Console.WriteLine("На сколько частей нужно поделить отрезок?");

N = Convert.ToDouble(Console.ReadLine());

j = AB / N;

Console.WriteLine(j);

x = a;

Console.Write("|" + "{0, 10}", "x" + "|");

Console.Write("{0, 10}", "y" + "|");

Console.Write("\n");

for (int i = 1; i < N; i++)

{

x += j;

if (x <= -4)

{

y = -2;

}

else if (x <= 0)

{

y = x / 4;

}

else if (x <= 2)

{

y = Math.Pow(x, 2);

}

else

{

y = (-x + 10) / 2;

}

Console.Write("|" + "{0, 10}", x + "|");

Console.Write("{0, 10}", y + "|");

Console.Write("\n");

}

}

}

}

}

## 4.3.Тестирование:

Контрольный пример 1:

Исходные данные:

a = -15;

b = 15;

N = 10;

Результат работы программы для указанных исходных данных приведён на рисунке 4.1

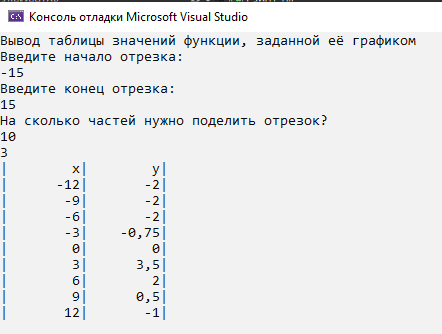


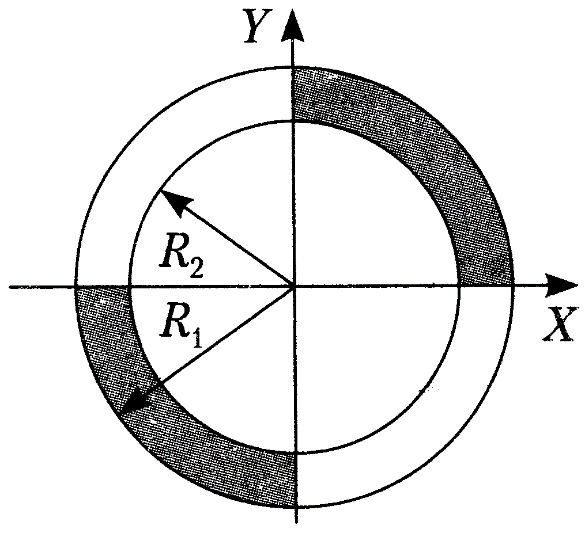
Рисунок 4.1 – Результат работы приложения Lab01\_04

# 5. Проверка серии «выстрелов по мишени»

5.1.Задание:Разработайте приложение, которое будет имитировать выполнение серии «выстрелов по мишени». Количество выстрелов и другие необходимые данные задаются пользователем. Для каждого «выстрела» необходимо выводить сообщение о попадании или промахе.

«Мишень» представляет собой область на плоскости, имеющую определённые геометрические размеры. «Выстрел» имитируется выбором некоторой точки на плоскости. Для каждого выстрела соответствующая точка на плоскости определяется случайным образом (в пределах прямоугольника или окружности, ограничивающих область вокруг мишени). «Попадание в мишень» засчитывается, если точка попадания принадлежит заданной области мишени.

Область задана графическим способом.

****

***Исходные данные:***

R1 – Радиус 1й (Большей) окружности;

R2 – Радиус 2й (Малой) окружности;

SH = Количество выстрелов;

***Результаты:***

Вывод пользователю сообщение о промахе или попадании в область, а также вывод случайных координат, имитирующих выстрел.

## 5.2.Код программы:

***Листинг 5 – Код программы Lab01\_05***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

Console.WriteLine("Добро пожаловать в программу имитирующую выстрелы по мишени");

double x, y, cord;

Console.WriteLine("Введите радиус первой (малой) окружности");

double R1 = Math.Abs(Convert.ToDouble(Console.ReadLine()));

Console.WriteLine("Введите радиус второй (большей) окружности");

double R2 = Math.Abs(Convert.ToDouble(Console.ReadLine()));

double k = 0;

Random rnd = new Random();

Console.WriteLine("Введите кол-во выстрелов:");

double SH = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Производим серию из " + SH + "случайных выстрелов...");

Console.Write("|" + "{0, 10}", "Выстрел №");

Console.Write("|" + "{0, 10}", "x" + "|");

Console.Write("{0, 10}", "y" + "| ");

Console.Write("\n");

for (int i = 1; i <= SH; i++)

{

x = rnd.Next(-1500, 1500);

x = Math.Round(x / 300, 4);

y = rnd.Next(-1500, 1500);

y = Math.Round(y / 300, 4);

if (((x <= 0) && (y <= 0)) || ((x >= 0) && (y >= 0)))

{

cord = x \* x + y \* y;

if (((R1 \* R1) <= cord) && (cord <= (R2 \* R2)))

{

Console.Write("|" + "{0, 10}", i);

Console.Write("|" + "{0, 10}", x + "|");

Console.Write("{0, 10}", y + "| ");

Console.Write("Попадание\n");

k++;

}

else

{

Console.Write("|" + "{0, 10}", i);

Console.Write("|" + "{0, 10}", x + "|");

Console.Write("{0, 10}", y + "| ");

Console.Write("Промах\n");

}

}

else

{

Console.Write("|" + "{0, 10}", i);

Console.Write("|" + "{0, 10}", x + "|");

Console.Write("{0, 10}", y + "| ");

Console.Write("Промах\n");

}

}

Console.WriteLine("Попаданий: " + k);

Console.WriteLine("Промахов: " + (SH - k));

}

}

}

## 5.3.Тестирование:

Контрольный пример 1:

Исходные данные:

R1 = 2;

R2 = 4;

SH = 10;

Результат работы программы для указанных исходных данных приведён на рисунке 5.1

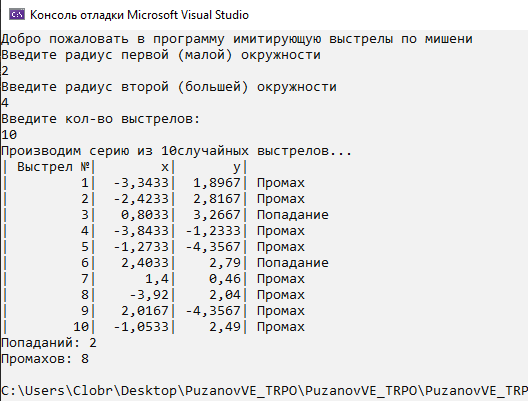


Рисунок 5.1 – Результат работы приложения Lab01\_05

# 6. Вычисление суммы ряда с заданной точностью

6.1.Задание:Разработайте приложение для вычисления суммы ряда Тейлора с заданной точностью . Все данные, необходимые для вычисления суммы ряда, вводятся пользователем.

Формула и область допустимых значений :



***Исходные данные:***

e – Точность задаваемая пользователем;

x – Вводимый х;

***Результаты:***

Вывод пользователю разложенный ряд Тейлора.

## 6.2.Код программы:

***Листинг 6 – Код программы Lab01\_06***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Добро пожаловать в программу, для вычисления суммы ряда Тейлора с заданной точностью");

Console.WriteLine("Введите погрешность измерений:");

double e = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите x");

double x = Convert.ToDouble(Console.ReadLine());

double F, S, Fact;

int i = 1;

F = 1;

Fact = 2;

Console.WriteLine("1й шаг - сумма равна " + F);

do

{

S = (Math.Pow(-1, i)) \* ((Math.Pow(x, (2 \* i)) / Fact));

F = F + S;

i++;

Fact = Fact \* ((2 \* i) - 1) \* (2 \* i);

Console.WriteLine(i + "й шаг - сумма равна " + F);

} while (Math.Abs(S) > e);

Console.WriteLine("Проверка:\nCos(" + x + ") = " + Math.Cos(x));

}

}

}

## 6.3.Тестирование:

Контрольный пример 1:

Исходные данные:

e = 0,0001

x = 1

Результат работы программы для указанных исходных данных приведён на рисунке 6.1

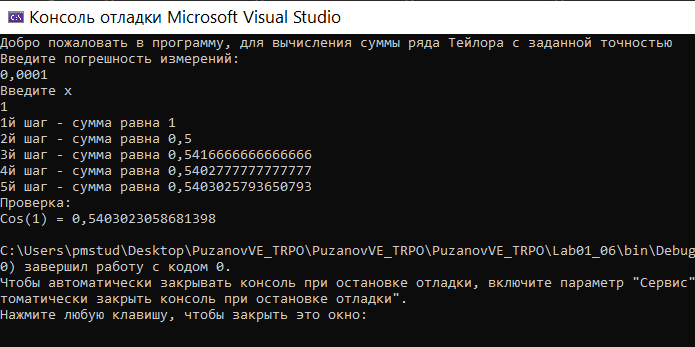


Рисунок 6.1 – Результат работы приложения Lab01\_06

Контрольный пример 2:

Исходные данные:

e = 0,0001

x = 15

Результат работы программы для указанных исходных данных приведён на рисунке 6.2

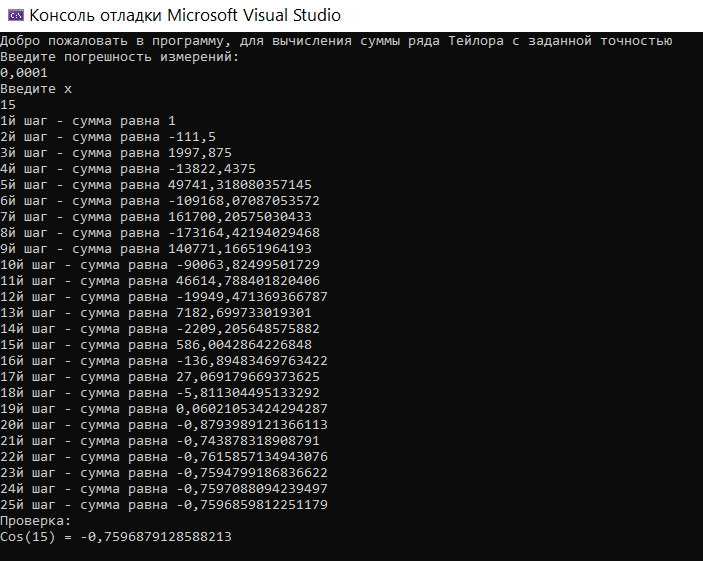


Рисунок 6.2 – Результат работы приложения Lab01\_06

# 7. Вывод таблицы значений функции, вычисляемой с помощью ряда Тейлора

7.1. Задание***:*** Разработайте приложение для вывода таблицы значений функции  в точках, полученных делением отрезка  на  равных частей. Функция  вычисляется как конечная сумма соответствующего ряда Тейлора с заданной пользователем точностью . Все данные, необходимые для вычисления значений функции и вывода таблицы, вводятся пользователем.

Формула и область допустимых значений :



***Исходные данные:***

a – начало отрезка;

b – конец отрезка;

N – Количество частей;

e – Точность задаваемая пользователем;

***Результаты:***

Вывод пользователю таблицы значений функции  в точках, полученных делением отрезка  на  равных частей.

## 7.2.Код приложения:

***Листинг 7 – Код приложения Lab01\_07***

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PuzanovVE\_TRPO

{

class Program

{

//14 Вариант

static void Main(string[] args)

{

Console.WriteLine("Добро пожаловать в программу для вычисления у");

double y, x, a, b, AB, N, j;

Console.WriteLine("Введите начало отрезка");

a = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Введите конец отрезка");

b = Convert.ToDouble(Console.ReadLine());

AB = b - a;

if (AB <= 0)

{

Console.WriteLine("Не существующий отрезок");

}

else

{

Console.WriteLine("На сколько частей нужно поделить отрезок?");

N = Convert.ToDouble(Console.ReadLine());

j = AB / N;

x = a;

Console.WriteLine("Введите погрешность измерений:");

double e = Convert.ToDouble(Console.ReadLine());

Console.Write("|" + "{0, 10}", "x" + "|");

Console.Write("{0, 22}", "y" + "|");

Console.Write("\n");

for (int k = 1; k < N; k++)

{

x += j;

double F, S, Fact;

int i = 1;

F = 1;

Fact = 2;

do

{

S = (Math.Pow(-1, i)) \* ((Math.Pow(x, (2 \* i)) / Fact));

F = F + S;

i++;

Fact = Fact \* ((2 \* i) - 1) \* (2 \* i);

} while (Math.Abs(S) > e);

Console.Write("|" + "{0, 10}", x + "|");

Console.Write("{0, 22}", F + "|");

Console.Write("\n");

}

}

}

}

}

## 7.3.Тестирование:

Контрольный пример 1:

Исходные данные:

a = -15;

b = 15;

N = 15;

e = 0,0001;

Результат работы программы для указанных исходных данных приведён на рисунке 7.1

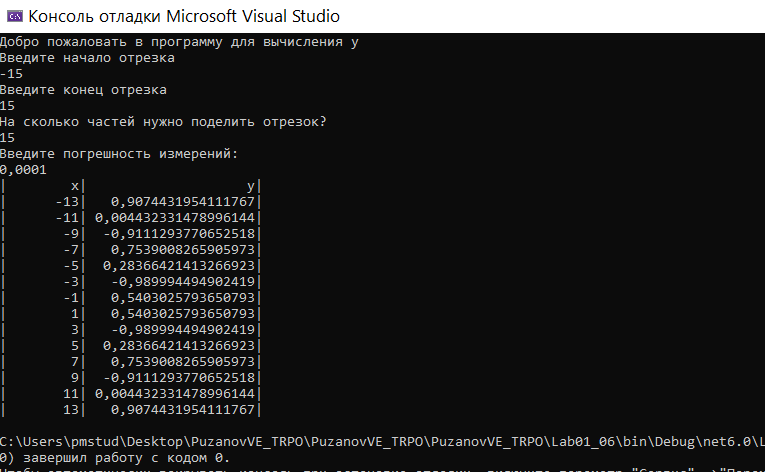


Рисунок 7.1 – Результат работы приложения Lab01\_07