Тема 2: «Вычисление функции с помощью разложения в ряд»

**Цель работы:** получить практические навыки разработки проектов для вычисления значения функции при помощи разложения в ряд (ряд Тейлора).

**Задание 1**

Необходимо разработать программу, вычисляющую значение функции с помощью разложения в ряд. Сумма ряда вычисляется при помощи цикла с неизвестным числом повторений, так как требуется найти значение с заданной точностью (допустимая погрешность вводится с клавиатуры). Сходящийся степенной ряд будет достигать искомого значения при достаточно большом количестве суммируемых членов ряда. При этом вклад последнего слагаемого в сумму будет пренебрежимо малым. Будем считать, что необходимая точность вычислений достигнута, когда относительный вклад последнего слагаемого в сумму по абсолютной величине станет меньше заданной погрешности (условие выхода из цикла накопления суммы). С целью повышения эффективности расчётов следует использовать рекуррентные формулы для выражения очередного члена ряда через предыдущий (чтобы избавиться от вычисления факториалов и больших степеней, вместо этого используйте умножение).

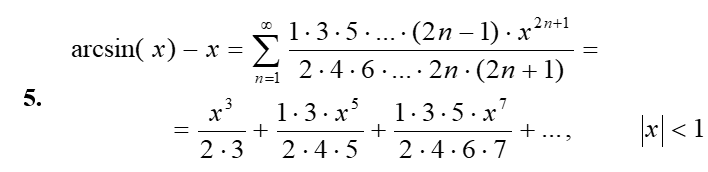
Для проверки полученного результата выведите значение указанной функции (левую часть выражения), вычисленную в цикле сумму ряда (правую часть выражения), а также количество просуммированных членов ряда.

**Задание 2**

Добавьте проверку:

* правильности ввода данных при нажатии клавиш в поля редактирования (используйте обработку события KeyPress);
* диапазона вводимых данных;
* контроль пустых полей ввода (заблокируйте кнопку «Вычислить», пока не будут введены все данные).

**Вариант 5**



Выполнение задания

1. **Математическая модель**

**Дано:**

x ϵ R, eps ϵ R(*x* – аргумент, *eps* - допустимая погрешность).

**Найти:**

Result, sum ϵ [-1;1]

– количество членов ряда

**Связь:**

(1) Result = arcsin(x)-x;

(2) sum =, где

(3) ;

(4) n: || ≤ eps∙||.

Вместо формул (2) и (3) используем рекуррентные формулы (2\*) –(5\*)

(2) (2\*)

(3) (3\*) , где

(5) ;

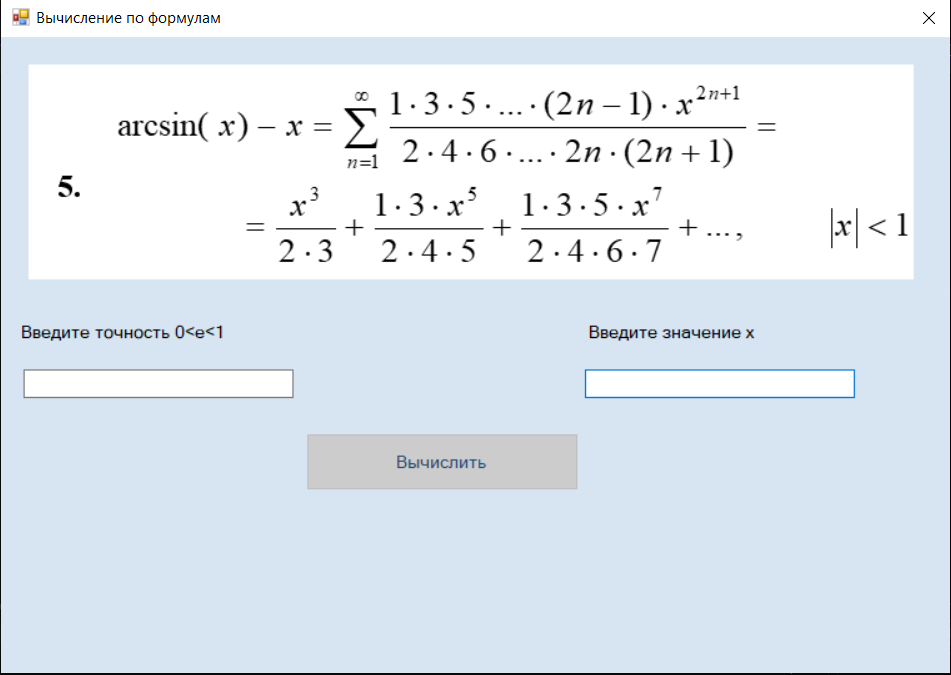
(5)  (5\*) .

**ОДЗ**:

0 ≤*x* <1, 0 < *eps* < 1

1. **Проектирование пользовательского интерфейса**

Внешний вид проекта представлен на рисунке 1, значения измененных свойств компонентов представлены в таблице 1.



label2

textBox2

label1

textBox1

button1

label\_res

Form1

pictureBox1

**Рис. 1.1**

1. **Свойства компонентов формы**

Значения свойств, установленные на этапе конструирования интерфейса:

*Таблица 1.*

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| form1 | Text | «Вычисление по формулам» |
| BackColor | GradientInactiveCaption |
| Font | Гарнитура: Microsoft Sans Serif |
| Размер: 10,2 |
| Начертание: обычный |
| label1 | Text | «Введите значение x» |
| label\_res | Text | «» |
| Font | Размер: 12 |
| FontColor | Brown |
| button1 | Text | «Вычислить» |
| Enabled | False |
| pictureBox1 | Image | team1.Properties.Resources.form |
| SizeMode | AutoSize |
| label2 | Text | «Введите точность 0<e<1» |

1. **Сценарий использования программы**
   1. Ввод x и eps в полях ввода.
   2. Нажать на кнопку Вычислить.
   3. Вычисление значений, result, sum и n - количество элементов, необходимых для вычисления значения sum.
   4. Возможность вернуться к пунктам 4.1 и/или 4.2.
   5. Завершение работы программы.
2. **Код программы**

**Код модуля Form1.cs:**

using ClassLibrary1;

using System;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace team1

{

public partial class Form1 : Form

{

Class1 cl;

public Form1()

{

InitializeComponent();

button1.Enabled = false;

cl = new Class1();

}

private void button1\_Click(object sender, EventArgs e)

{

double z1;

double z2;

try

{

if (double.Parse(textBox1.Text) < 1&& double.Parse(textBox2.Text)>0 && double.Parse(textBox2.Text) < 1)

{

double x = double.Parse(textBox1.Text);

double eps = double.Parse(textBox2.Text);

z1 = cl.calculate(x, eps);

z2 = cl.calculate\_arcsin(x, eps);

label\_res.Text = string.Format("Ответ:\r\n Сумма ряда = {0} \r\n arcsin(x)-x = {1}\r\n Количество членов ряда {2}", z1, z2, cl.n);

}

else

{

textBox1.Text = null;

textBox2.Text = null;

label\_res.Text = "Выход за диапазон допустимых значений!";

return;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.Text.Length > 0)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

if (textBox2.Text.Length > 0)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

if (textBox1.Text.StartsWith(","))

{

button1.Enabled = false;

}

if (textBox1.Text == "-")

{

button1.Enabled = false;

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

bool hasComm = (sender as TextBox).Text.Contains(',');

bool hasMin = (sender as TextBox).Text.Contains('-');

if ((e.KeyChar == ',' && hasComm) || (e.KeyChar == '-' && hasMin))

{

e.Handled = true;

}

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != ',') && (e.KeyChar != '-'))

{

e.Handled = true;

}

if ((e.KeyChar == ',' && hasComm) || (e.KeyChar == '-' && hasMin))

{

e.Handled = true;

}

if (e.KeyChar == '-' && hasComm)

{

e.Handled = true;

}

if (e.KeyChar == ',' && hasMin&&textBox1.Text.Length>2)

{

e.Handled = true;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

if (textBox1.Text.Length > 0)

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

if (textBox2.Text.StartsWith(","))

{

button1.Enabled = false;

}

if (textBox2.Text=="-")

{

button1.Enabled = false;

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != ',') && (e.KeyChar != '-'))

{

e.Handled = true;

}

bool hasComm = (sender as TextBox).Text.Contains(',');

bool hasMin = (sender as TextBox).Text.Contains('-');

if ((e.KeyChar == ',' && hasComm) || (e.KeyChar == '-' && hasMin))

{

e.Handled = true;

}

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar) && (e.KeyChar != ',') && (e.KeyChar != '-'))

{

e.Handled = true;

}

if (e.KeyChar == '-')

{

e.Handled = true;

}

}

private void label\_res\_MouseEnter(object sender, EventArgs e)

{

label\_res.BackColor = Color.Aqua;

}

private void label\_res\_MouseLeave(object sender, EventArgs e)

{

label\_res.BackColor = SystemColors.GradientInactiveCaption;

}

}

}

**Код модуля ClassLibrary1.dll:**

using System;

namespace ClassLibrary1

{

public class Class1

{

public Class1() { }

public int n;

public double calculate(double x, double epsilon)

{

try

{

double term = x;

double b = x;

double result =0;

double par = x \* x / 2;

n = 1;

while (Math.Abs(term) >epsilon\* Math.Abs(result))

{

b\* = par\* (2 \* n - 1) / n;

term = b/(2\*n+1);

result += term;

n++;

}

return result;

}

catch (Exception e)

{

throw new Exception(e.ToString());

}

}

public double calculate\_arcsin(double x, double epsilon)

{

double result = 0;

result = Math.Asin(x) - x;

return result ;

}

}

}

1. **Схемы основных подпрограмм**



**Рис 6.1.** **Обработка ввода textBox\_KeyPress**

x < 1&& eps>0

&& eps < 1

double x = double.Parse(textBox1.Text);

double eps = double.Parse(textBox2.Text);

да

нет

Вывод(ln, sum, n\_count)

Z1 = calculate(x,eps)

Z2 = calculate\_arcsin(x,eps)

Начало

Конец

Вывод(“Выход за диапазон допустимых значений”)

**Рис 6.2. Метод button1\_Click**

Math.Abs(term) >epsilon\* Math.Abs(result)

Ввод x, epsilon

double term = x; double b = x; double result = 0; double par = x \* x / 2; n = 1;

да

нет

Вывод(result)

Начало

Конец

b = par\* (2 \* n - 1) / n; term \*= b/(2\*n+1);

result += term;

n++

да

**Рис 6.3. Метод calculate**

Ввод x, epsilon

double result = 0;

Вывод(result)

Начало

Конец

result = Math.Asin(x) - x;

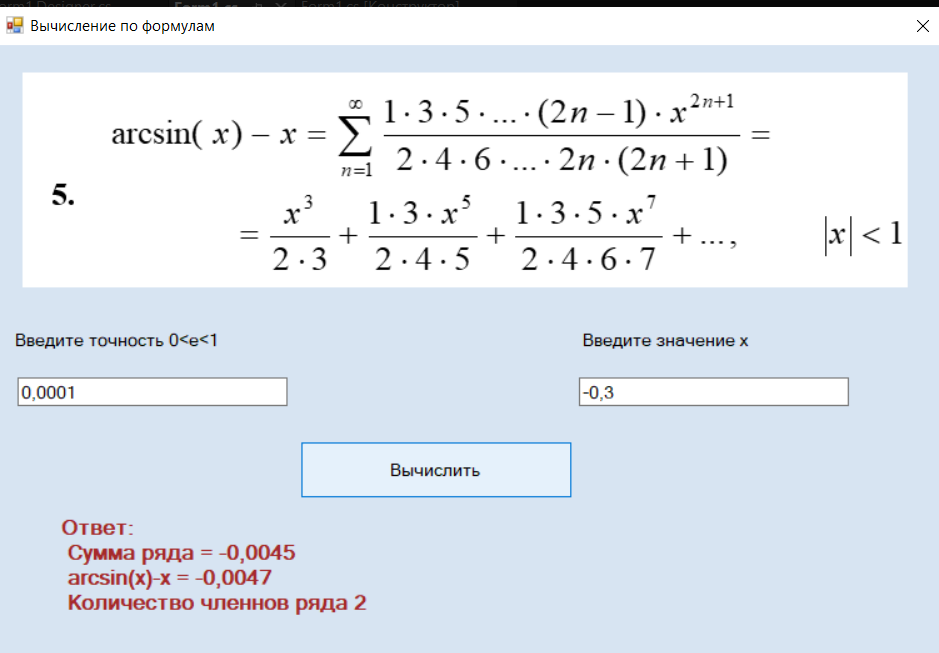
**Рис 6.2. Метод calculate\_arcsin**

1. **Протокол испытаний**

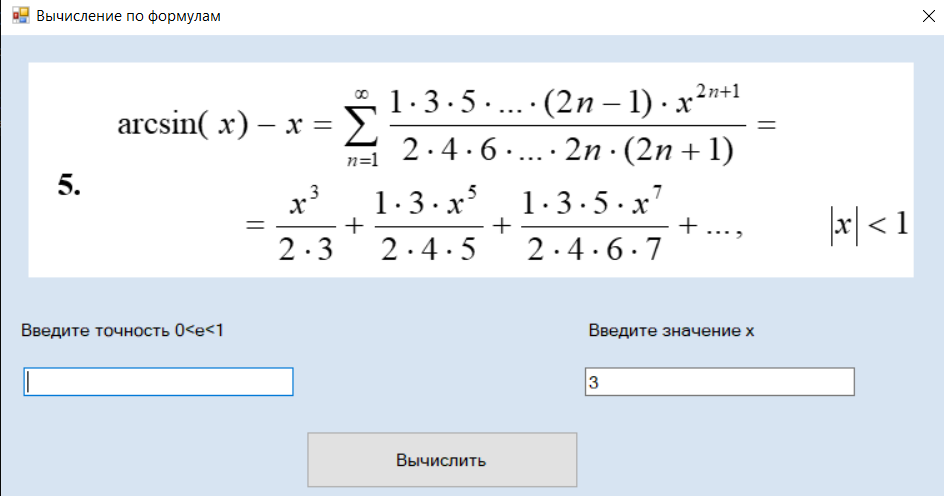
*Таблица 2.* Протокол испытаний

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность ввода данных | e = 0,0001; x = -0,3 | Числа вводятся в текстовое поле | Такие же, как ожидаемые результаты  (См. Рис. 6) |
| Способность контроля вводимых данных | e = fdf; x = 3ld | e =””  x = “3” | Такие же, как ожидаемые результаты  (См. Рис. 7) |
| Способность программы обеспечить контроль вводимых данных. | e = 0; x = 0,3- | e = 0  x = 0,3 | Соответствуют ожиданиям.  (См. Рис. 8) |
| Способность контроля непереводимых полей ввода | Оставить поле ввода пустым или с знаками ‘-‘ и ‘,’ (без чисел) | Кнопка для решения заблокирована | Такие же, как ожидаемые результаты  (См. Рис.9) |
| Способность контроля недопустимых значений e и x | Входные данные: e = 6; x = 56; | Оповещение о вводе некорректных данных | См.Рис.10 |
| Проверить способность программы вычислять значения ln(x+1) по рекурсивной формуле | Ввод значений e и x и нажатие кнопки «Вычислить»  Входные данные 1: e = 0,00001; x = 0,1  Входные данные 2: e = 0,0001; x = 0,2 | Результат 1: arcsin(x)-x=0,00016742 Сумма = 0,0001669  Результат 2: arcsin(x)-x=0,001357920 Сумма = 0,001341371 | См.Рис. 11, 12 |

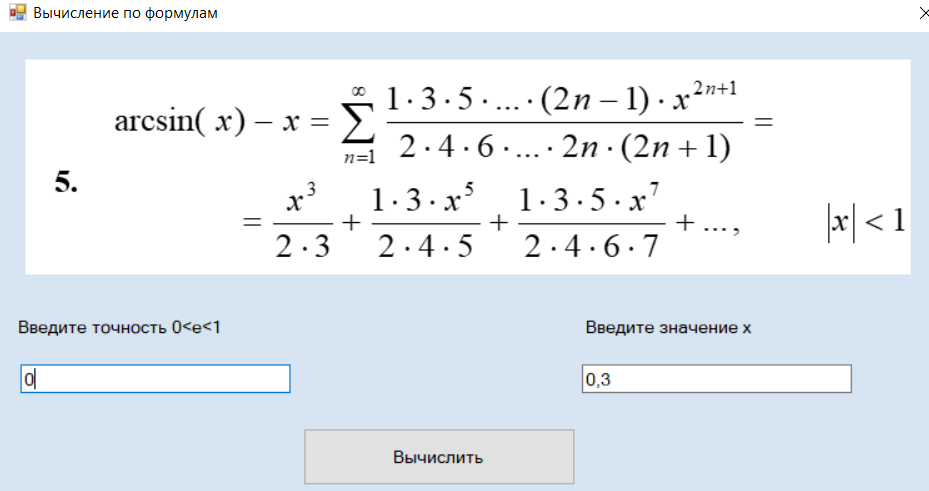
1. **Тестирование проекта:**



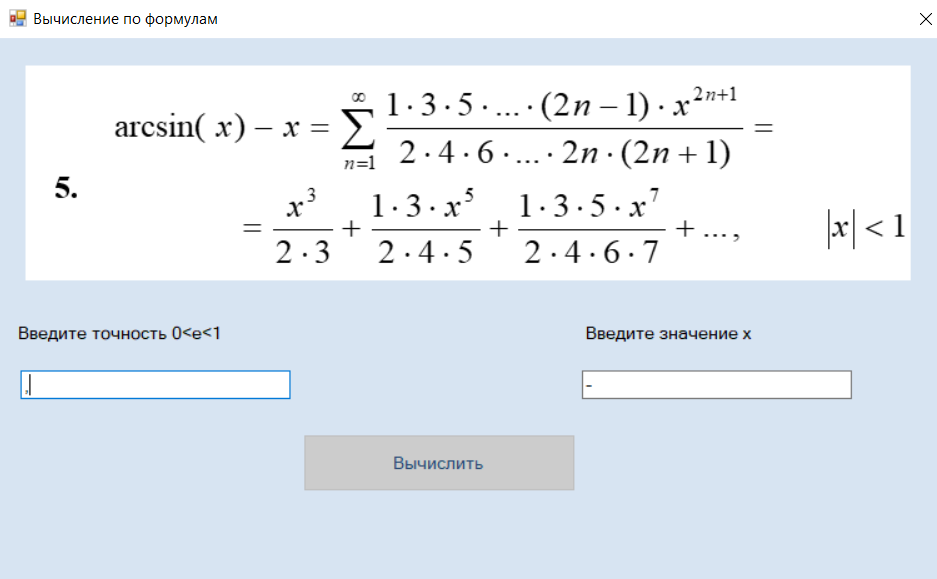
**Рис. 6**

****

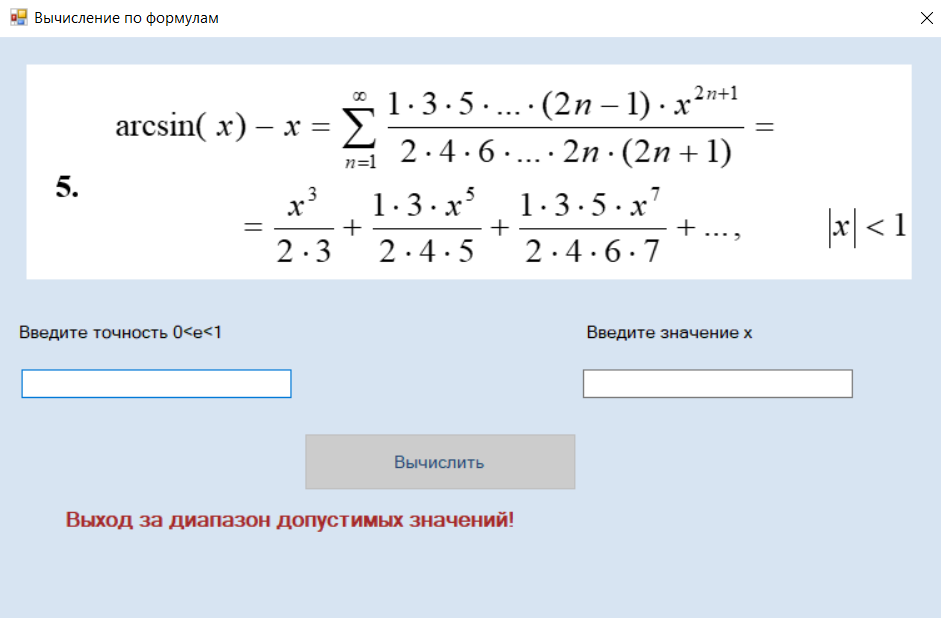
**Рис. 7**

****

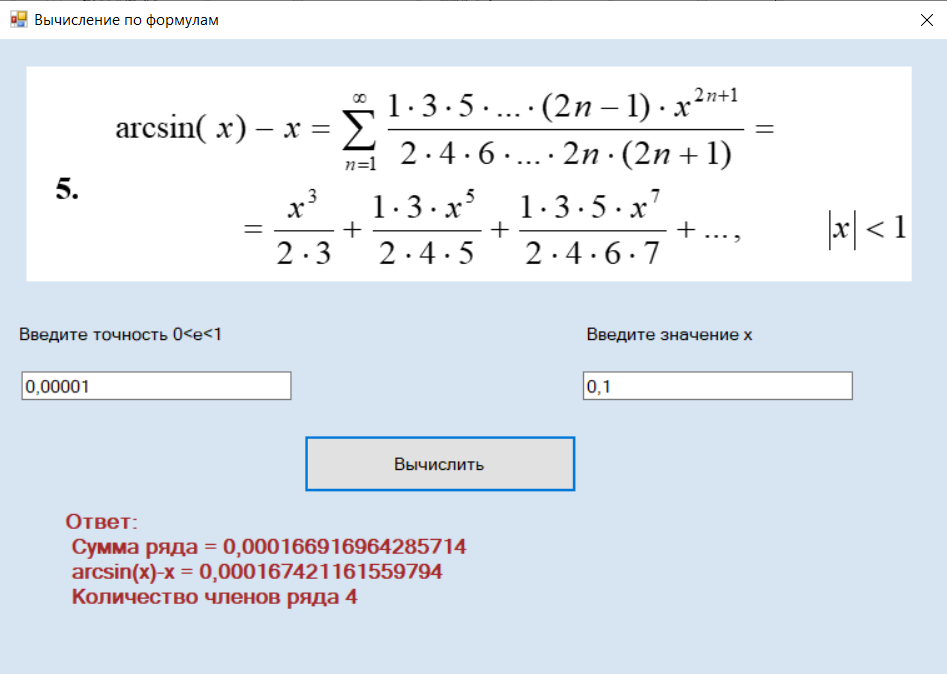
**Рис. 8**



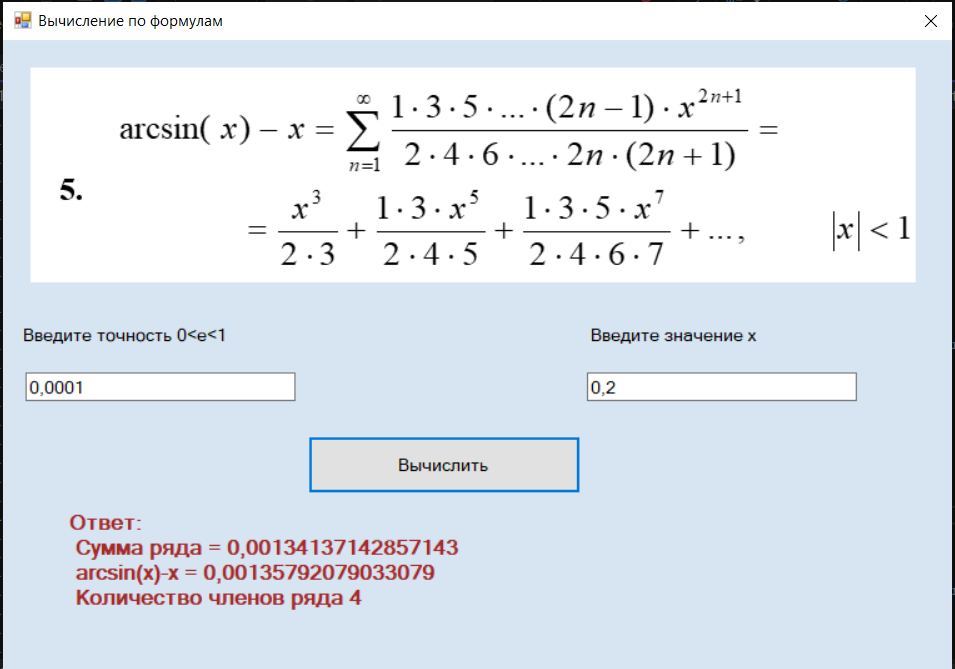
**Рис. 9**

****

**Рис. 10**



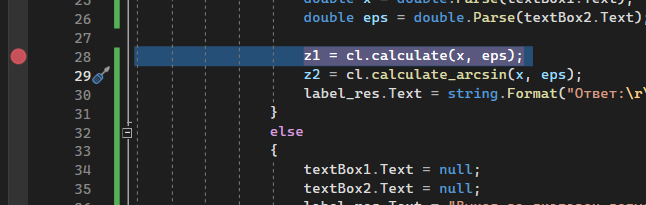
**Рис. 11**

****

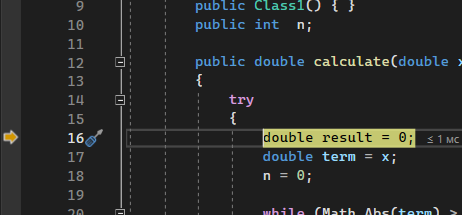
**Рис. 12**

**Задание 3.** **Работа с отладчиком**

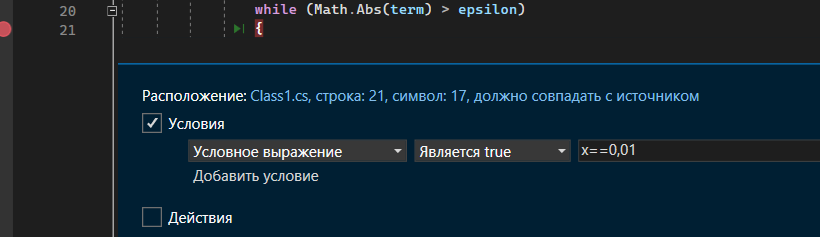
Выполнение задания 3 представлено на рисунках 3.1.1-3.2.3



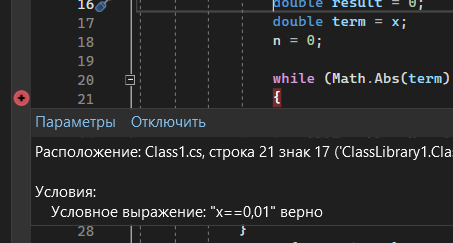
**Рис. 3.1.1. Постановка точки останова**



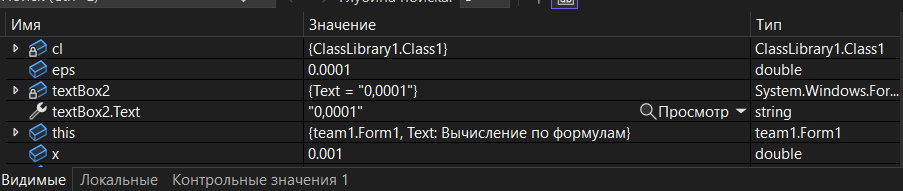
**Рис. 3.1.2. Пошаговая отладка**



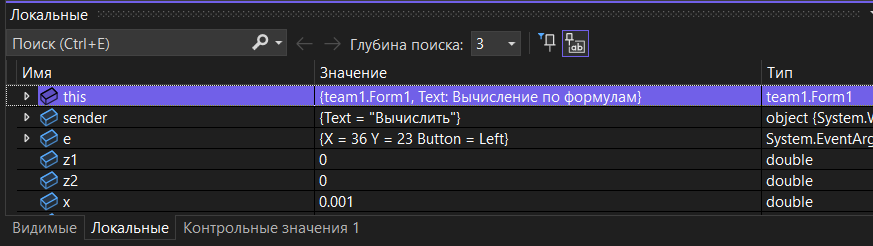
**Рис. 3.1.3. Точка останова с условием**



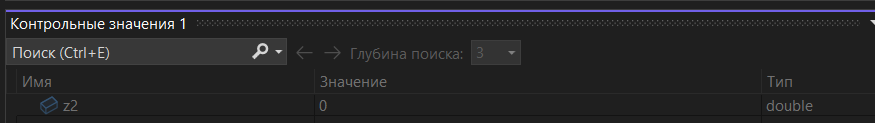
**Рис. 3.1.4. Отработанная точка останова с условием**



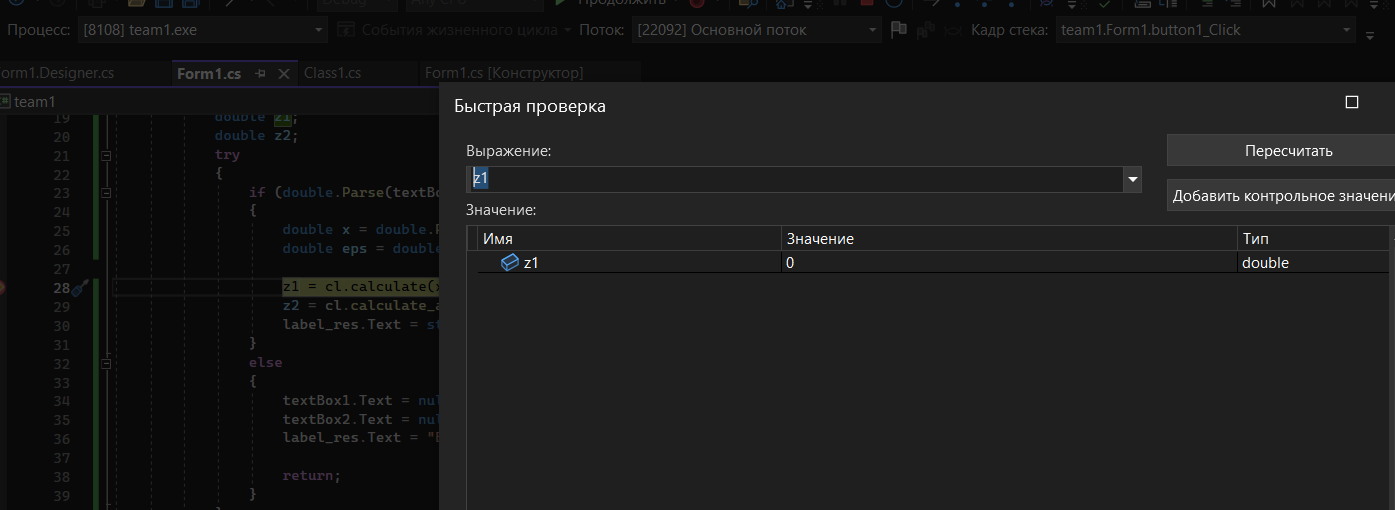
**Рис. 3.1.5. Окно «Видимые» значения переменных**



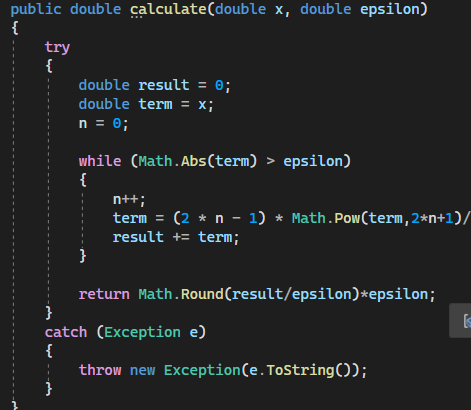
**Рис. 3.1.6. Окно «Локальные» значения переменных**



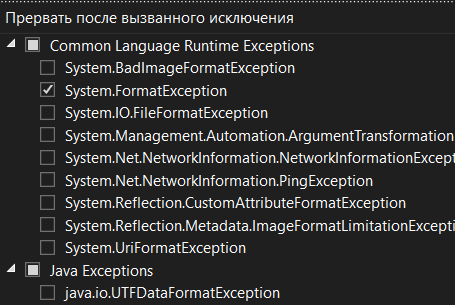
**Рис. 3.1.7. Работа окна «Контрольные значения»**



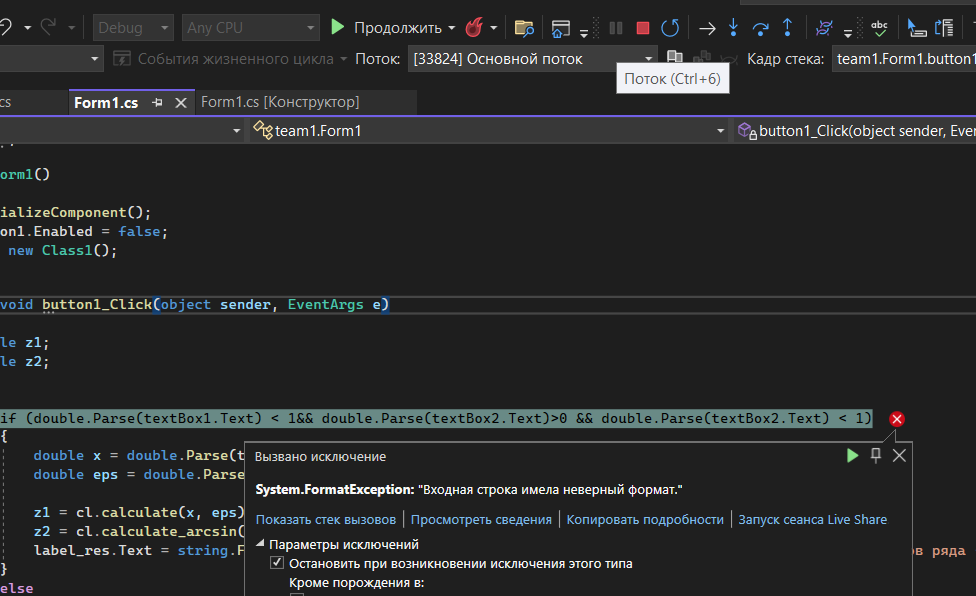
**Рис. 3.1.8. Работа окна «Быстрая проверка»**



**Рис. 3.2.1. Добавление в код обработчика исключений**

****

**Рис. 3.2.2. Настройка отладчика при срабатывании исключения**

****

**Рис. 3.2.3. Результат настройки**