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).

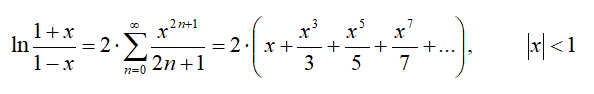
Для проверки полученного результата выведите значение указанной функции (левую часть выражения), вычисленную в цикле сумму ряда (правую часть выражения), а также количество просуммированных членов ряда.

***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 3**



Выполнение задания

1. **Математическая модель**

**Дано:**

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

**Найти:**

**Связь:**

(1) Ln = ln(1+x/1-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**:

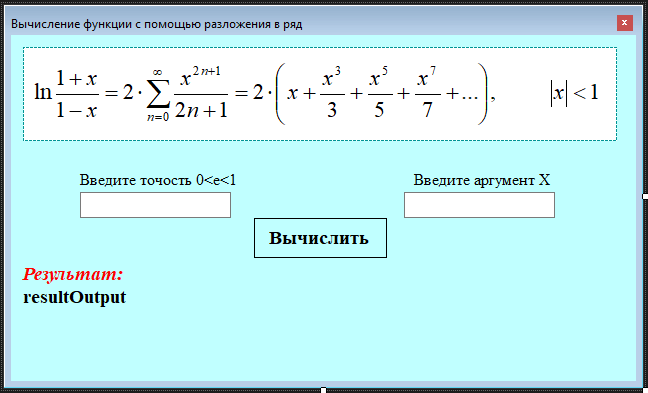
|*x|* <1, 0 <eps<1

1-x ≠ 0

1. **Проектирование пользовательского интерфейса**

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

Form1



resultOutput

label2

textBox2

Label3

Button1

textBox1

label1

pictureBox1

**Рис. 2.1**

1. **Свойства компонентов формы**

*Таблица 1.* Значения свойств, установленные на этапе конструирования интерфейса

|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| form1 | Text | «Вычисление функции с помощью разложения в ряд» |
| Font | Microsoft Sans Serif 8pt |
| MaximizeBox | False |
| FormBorderStyle | FixedSingle |
| label1 | Text | «Введите точность 0 < e < 1» |
| Font | Times New Roman 12pt |
| label2 | Text | «Введите аргумент Х» |
| Font | Times New Roman 12pt |
| textBox1 | MaxLength | 7 |
| Font | Times New Roman 12pt |
| textBox2 | MaxLength | 7 |
| Font | Times New Roman 12pt |
| Label3 | Text | «Результат:» |
| Font | Times New Roman 14 Bold, Italic |
| FontColor | Red |
| Visible | False |
| resultOutput | Font | Times New Roman 14 Bold |
| FontColor | Control |
| Visible | False |
| Button1 | Font | Times New Roman 14 Bold |
| Text | «Вычислить» |
| Enabled | False |
| pictureBox1 | Image | PR\_2.Properties.Resources.Prakt2 |
| SizeMode | AutoSize |

1. **Сценарий использования программы**
   1. Ввод x и eps в полях ввода.
   2. Нажатие на кнопку Вычислить.
   3. Вычисление значений Ln, 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 System;

using System.Linq;

using System.Windows.Forms;

namespace Prakt2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

label3.Visible = false;

resultOutput.Visible = false;

button1.Enabled = false;

}

private void Sum(double e, double x)

{

double sum = x;

double ln = Math.Log((1 + x) / (1 - x));

double nplus = 0;

double n = x;

int count = 0;

for (int i = 1; Math.Abs(nplus) <= e \* sum && count != 1000; i++)

{

nplus = n \* (x \* x \* (2 \* i - 1) / (2 \* i + 1));

sum += nplus;

count++;

}

sum \*= 2;

if (count == 1000)

{

label3.Visible = true;

resultOutput.Visible = true;

resultOutput.Text = string.Format("ln((1 + x) / (1 - x)) = {0:G7}\n", ln);

resultOutput.Text += string.Format("Сумма ряда не найдена из-за большого\n количества итераций", sum);

}

else

{

label3.Visible = true;

resultOutput.Visible = true;

resultOutput.Text = string.Format("ln((1 + x) / (1 - x)) = {0:G7}\n", ln);

resultOutput.Text += string.Format("Сумма ряда: {0:G7}\n", sum);

resultOutput.Text += string.Format("Кол-во элементов в ряде: {0:G7}", count);

}

}

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

{

double x = Convert.ToDouble(textBox2.Text);

double eps = Convert.ToDouble(textBox1.Text);

if ((eps > 0) && (eps < 1)){

if (Math.Abs(x) < 1)

{

Sum(eps, x);

}

else

{

resultOutput.Text = "значение X не удовлетворяет ОДЗ";

resultOutput.Visible = true;

}

}

else

{

resultOutput.Text = "значение eps не удовлетворяет ОДЗ";

resultOutput.Visible = true;

}

}

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

{

if (textBox2.Text != "" && textBox2.TextLength != 0 && textBox1.Text != "" && textBox1.TextLength != 0)

{

if (textBox2.Text[textBox2.TextLength - 1] != ',' && textBox1.Text[textBox1.TextLength - 1] != ',' && textBox2.Text != "-")

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

}

else

{

button1.Enabled = false;

}

}

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

{

if (textBox2.Text != "" && textBox2.TextLength != 0 && textBox1.Text != "" && textBox1.TextLength != 0)

{

if (textBox2.Text[textBox2.TextLength - 1] != ',' && textBox1.Text[textBox1.TextLength - 1] != ',' && textBox2.Text != "-")

{

button1.Enabled = true;

}

else

{

button1.Enabled = false;

}

}

else

{

button1.Enabled = false;

}

}

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

{

if (c.KeyChar != ' ' && c.KeyChar == '\b' || (Char.IsNumber(c.KeyChar)

|| (!textBox1.Text.Contains(',') && c.KeyChar == ',' && textBox1.TextLength >= 1)))

{

}

else

{

c.Handled = true;

}

}

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

{

if (e.KeyChar != ' ' && e.KeyChar == '\b'

|| (Char.IsNumber(e.KeyChar) || (e.KeyChar == '-' && textBox2.TextLength == 0)

|| (!textBox2.Text.Contains(',') && e.KeyChar == ',' && textBox2.TextLength >= 1 && !textBox2.Text.Contains('-'))

|| (!textBox2.Text.Contains(',') && e.KeyChar == ',' && textBox2.TextLength >= 2 && textBox2.Text.Contains('-'))))

{

}

else

{

e.Handled = true;

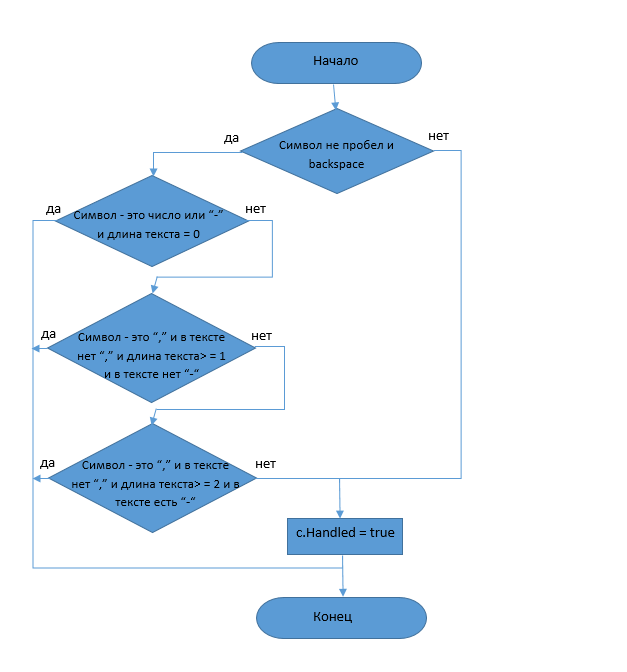
}

}

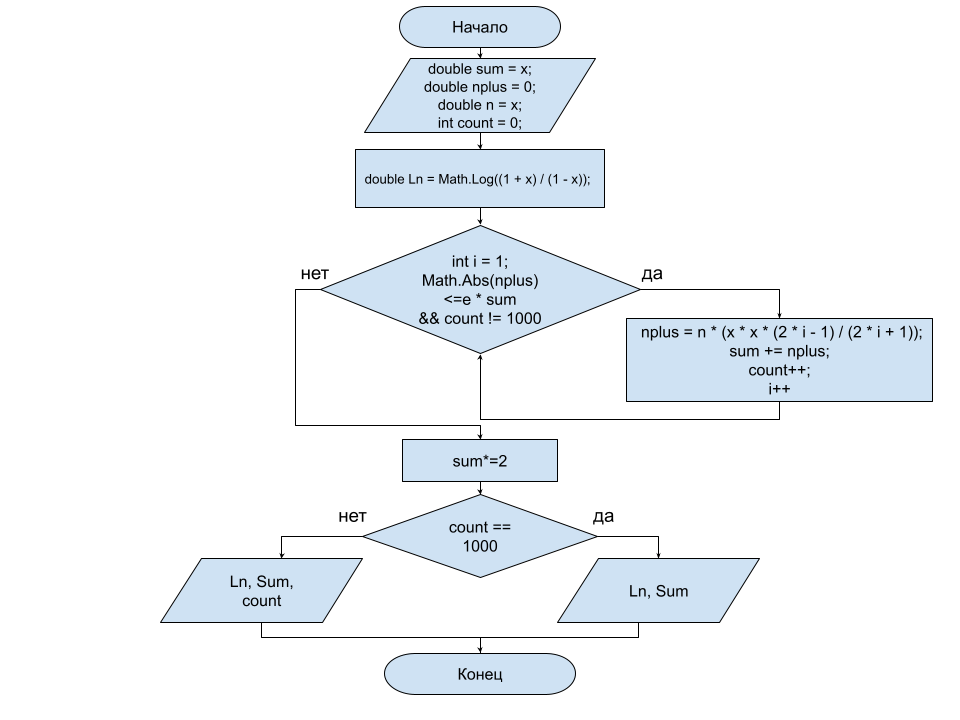
}

}

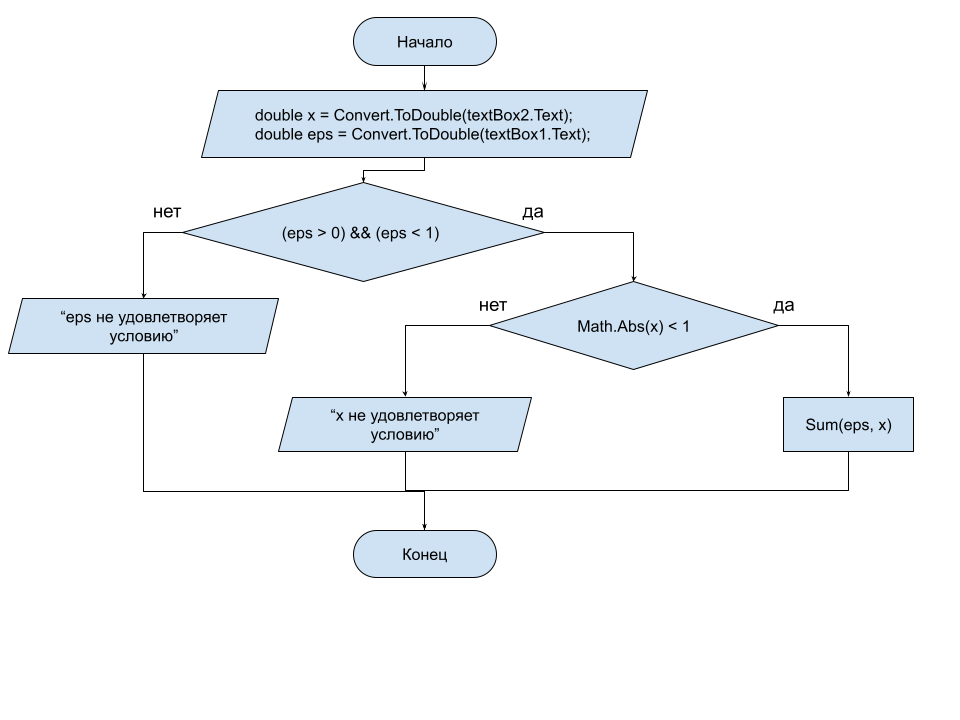
1. **Схемы основных подпрограмм**



**Рис. 6.1 Схема обработки ввода KeyPress для полей textBox1 и textBox2**



**Рис. 6.2 Метод Sum**



**Рис. 6.3 Метод buttonResult\_Click**

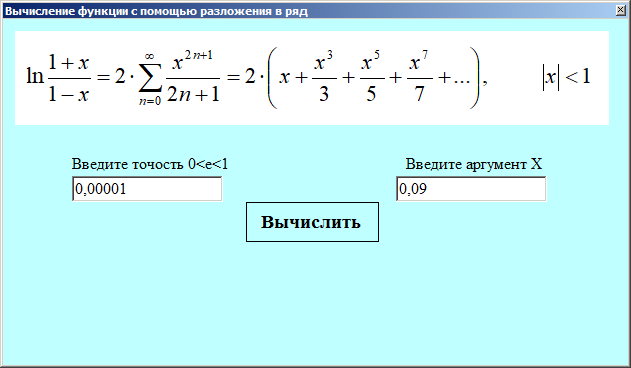
1. **Протокол испытаний**

*Таблица 2.* Протокол испытаний

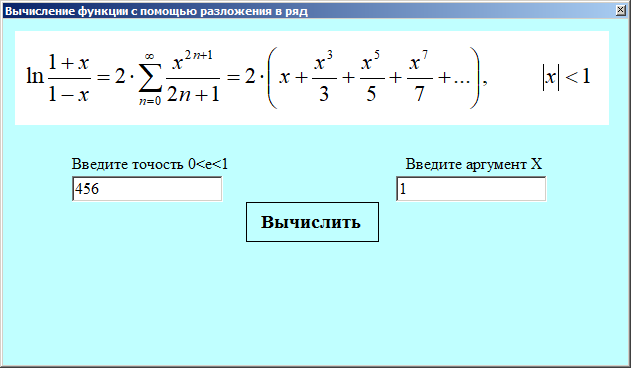
|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность ввода данных | e = 0,00001; x = 0,09 | Числа вводятся в текстовое поле | Такие же, как ожидаемые результаты  (См. Рисунок 8.1) |
| Способность контроля вводимых данных | e =EA456; x = AKF1R | e = 456  x = 1 | Такие же, как ожидаемые результаты  (См. Рисунок 8.2) |
| Способность программы обеспечить контроль вводимых данных. | e = -0-; x = 0,3- | e = 0  x = 0,3 | Соответствуют ожиданиям.  (См. Рисунок 8.3) |

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность контроля непереводимых полей ввода | Оставить поля ввода пустым или с знаками ‘-‘ и ‘,’ (без чисел) | Кнопка для решения заблокирована | Такие же, как ожидаемые результаты (См. Рисунок 8.4) |
| Способность контроля недопустимых значений eps | Входные данные: e = 1; x = 0,1; | Оповещение о вводе некорректных данных | Такие же, как ожидаемые результаты (См. Рисунок 8.5) |
| Способность контроля недопустимых значений X | Входные данные: e = 0,1; x = -3; | Оповещение о вводе некорректных данных | Такие же, как ожидаемые результаты (См. Рисунок 8.6) |
| Проверить способность программы вычислять значения ln((1 + x) / (1 - x)) по рекурсивной формуле | Ввод значений e и x и нажатие кнопки «Вычислить»  Входные данные 1: e = 0,00001; x = 0,6  Входные данные 2: e = 0,001; x = 0,1 | Результат 1: ln((1 + x) / (1 - x))=1,386294 Sum = 1,344  Count = 1  Результат 2: ln((1 + x) / (1 - x))=0,2006707 Sum = 0,2006667  Count = 1 | Такие же, как ожидаемые результаты (См. Рисунок 8.7 и Рисунок 8.8) |

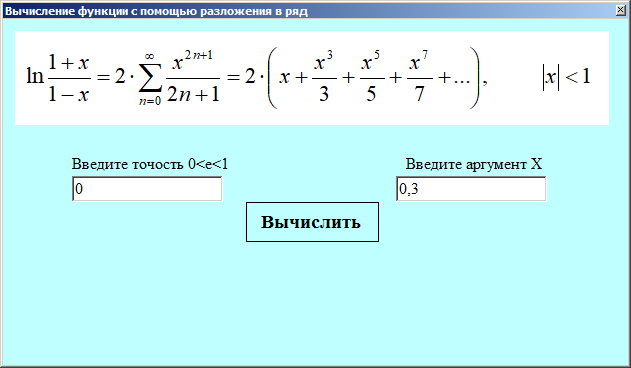
1. **Тестирование проекта:**



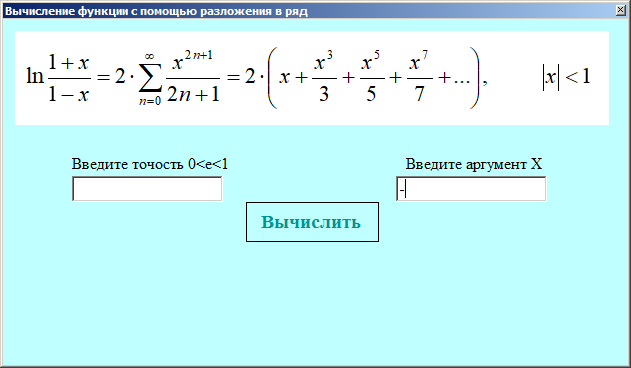
**Рис. 8.1**



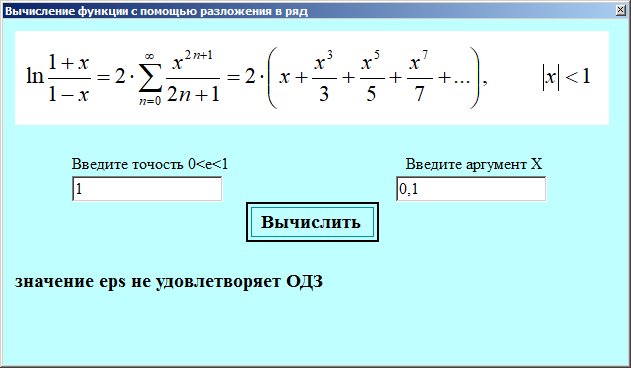
**Рис. 8.2**



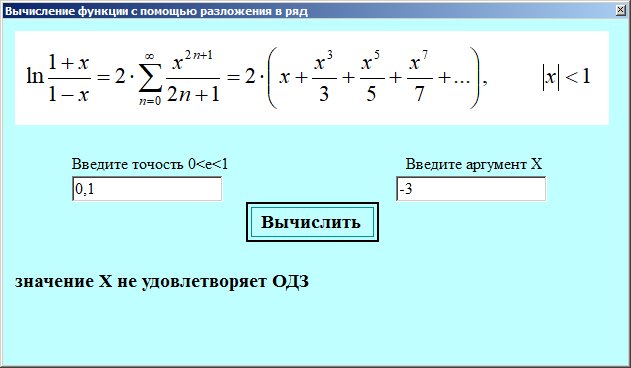
**Рис. 8.3**



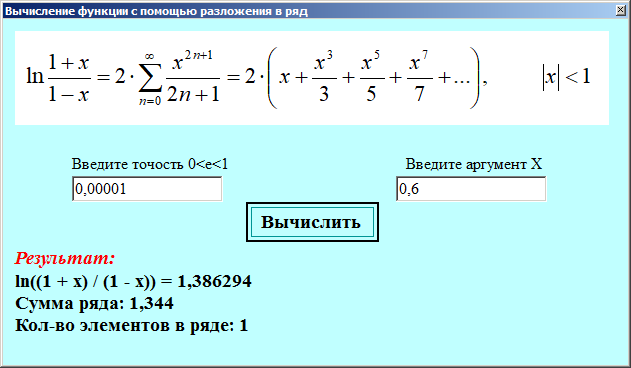
**Рис. 8.4**



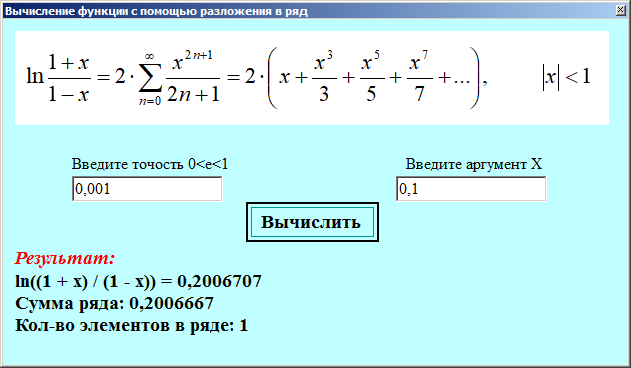
**Рис. 8.5**



**Рис. 8.6**



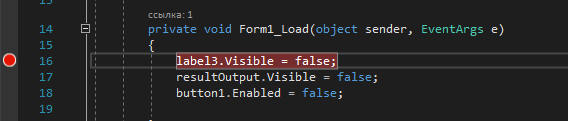
**Рис. 8.7**



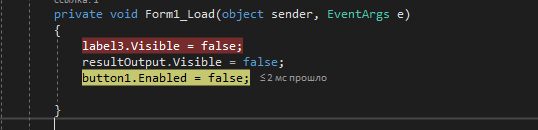
**Рис. 8.8**

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

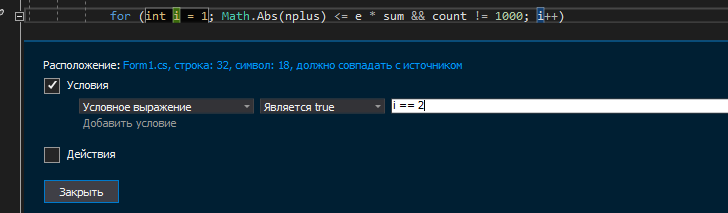
Выполнение задания 3 представлено на рисунках 3.1.1-3.2.3



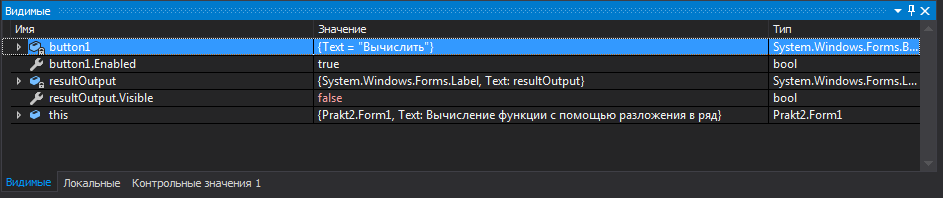
**Рис. 3.1.1 Точка останова**



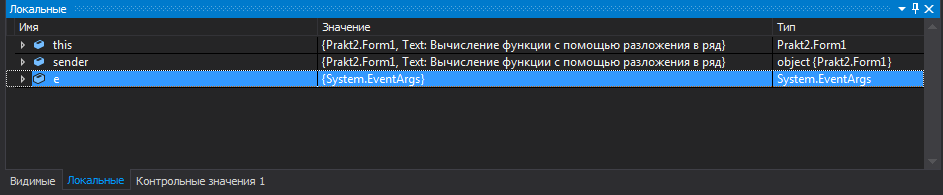
**Рис. 3.1.2 Выполнение команд пошагово**



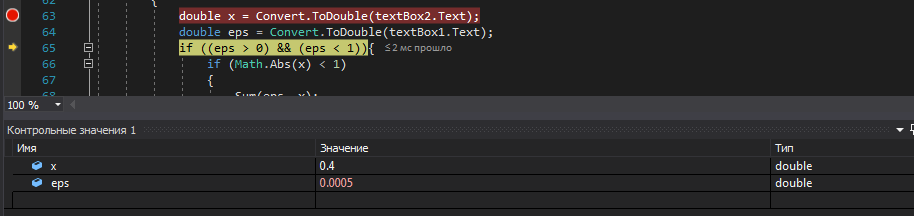
**Рис. 3.1.3 Точка останова с условием в цикле**



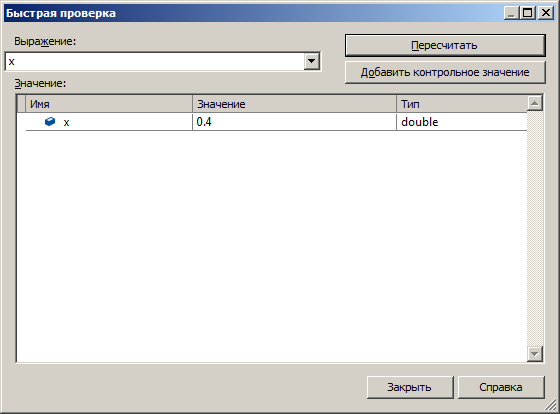
**Рис. 3.1.4 Видимые переменные**



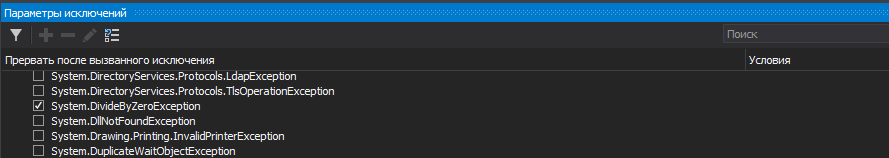
**Рис.3.1.5 Локальные переменные**



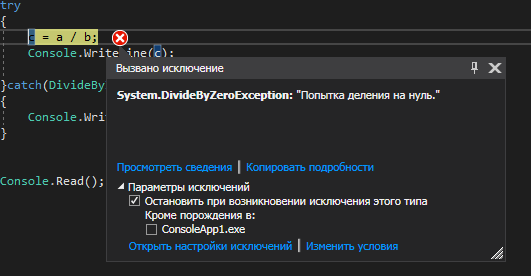
**Рис. 3.1.6 Контрольные значения**



**Рис. 3.1.8 Окно быстрой проверки**



**Рис. 3.2.1 Добавление исключения**



**Рис. 3.2.2 Проверка управления исключениями с помощью отладчика**