**Тема 1:** Вычисление функции с помощью разложения в ряд

**Цель работы:** получить практические навыки разработки проектов для вычисления

значения функции при помощи разложения в ряд (рядов Тейлора). Отработать

навыки работы с отладчиком.

Математическая формулировка задачи

Дано:

X – аргумент для вычисления

Одз: 0 < eps < 1

Найти

S – сумма ряда (тип double)

F – Math.Arctan(x) – математическая функция (тип Double)

N – количество членов рядя (тип int32)

Связь

U – текущий член ряда (тип Double)

Up – предыдущий член ряда

S – сумма ряда (тип Double)

n – количество членов рядя (int32)

coef – Коэффициент который надо прибавлять для получения знаменателя (int32)

znam – знаменатель члена ряда

Начальные условия

U = 1/x; Up = 0; n=0; S=0; coef=0;

На каждом проходе цикла:

Up = U;

S += U;

n++;

znam = (2 + coef + 1) / (coef + 1) \* (x \* x)

U = -1 \* (U \* (1/znam);

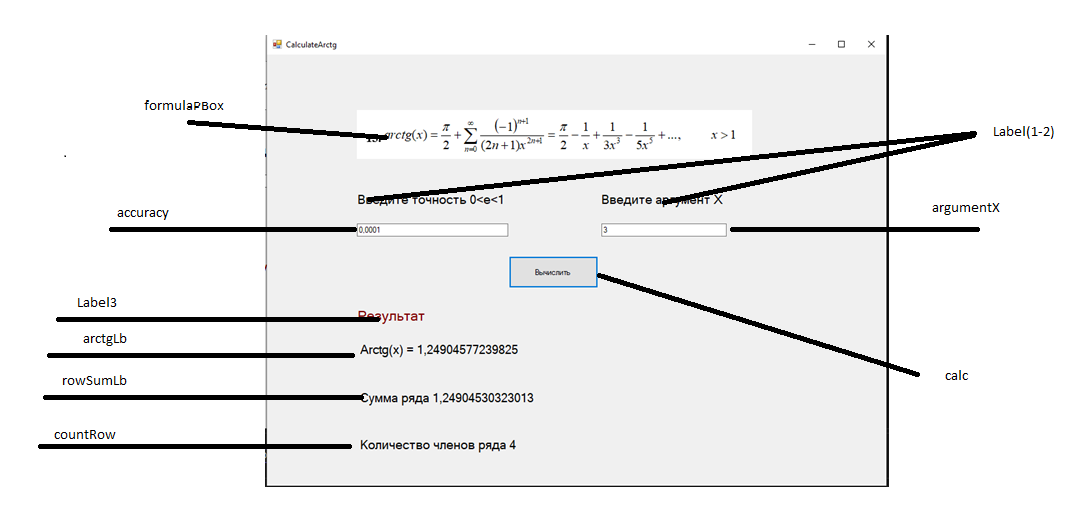
Условие прохождения цикла |U-Up| > eps

S += - после выхода из цикла

Функциональные характеристики программы

1. Обеспечивать ввод исходных данных с клавиатуры
2. Контроль исходных данных на валидность и ОДЗ
3. Вычисление суммы ряда и вывод на экран функции (левая часть выражения),

Суммы ряда (правая часть выражения) и количество просуммированных членов ряда



|  |  |  |
| --- | --- | --- |
| **Название компонента** | **Свойства** | **Значения** |
| formulaPBox | SizeMod | StretchImage |
| label1 | Text | «Введите точность 0 < e < 1» |
| label2 | Text | «Введите аргумент X» |
| accuracy/argumentX | Text | «» |
| calc | Text | «Вычислить» |
| Label3 | Text | «Результат» |
| ForeColor | Maroon |
| arctgLb | Text | «Arctg(x) = » |
|  | ForeColor | black |
| rowSumLb | Text | «Сумма ряда » |
|  | ForeColor | black |
| countRow | Text | «Количество членов ряда » |
|  | ForeColor | black |

Листинг кода

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

int rowCount;

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

{

calc.Enabled = false;

}

const int ODZ = 1;

const int LEFT\_BORDERS = 0;

const int RIGHT\_BORDERS = 1;

/// <summary>

/// Подсчет суммы ряда и количества элементов ряда

/// </summary>

/// <param name="x"></param>

/// <param name="e"></param>

/// <param name="rowLong"></param>

/// <returns></returns>

private double rowSum(double x, double e, out int rowLong)

{

double U = -(1 / x);

double Up;

rowLong = 0;

double S = U;

int coef = 0;

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

{

Up = U;

rowLong++;

double znam = (double)(2 + coef + 1) / (coef + 1) \* (x \* x);

U = -1 \* (U \* (1 / znam));

coef += 2;

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

S += U;

if (!(Math.Abs(U - Up) > e)) return (Math.PI/2) + S;

}

}

/// <summary>

/// Event на нажатие кнопки, вызывает функцию rowSum

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

double argX = 0;

double acc = 0;

try

{

argX = double.Parse(argumentX.Text);

acc = double.Parse(accuracy.Text);

if (!(argX > ODZ)) throw new Exception("Аргумент не подходит под одз");

if(acc<=LEFT\_BORDERS || acc>=RIGHT\_BORDERS) throw new Exception("Точность введена неверно");

double sum = rowSum(double.Parse(argumentX.Text), double.Parse(accuracy.Text), out rowCount);

arctgLb.Text = $"Arctg(x) = {Math.Atan(double.Parse(argumentX.Text))}";

rowSumLb.Text = $"Сумма ряда {sum}";

countRow.Text = $"Количество членов ряда {rowCount}";

}

catch(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Ограничение на ввод только чисел и ','

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

var txb = sender as TextBox;

if (txb.Text.Contains(",") && e.KeyChar == ',') { e.Handled = true; return; }

if (!char.IsDigit(e.KeyChar) && e.KeyChar != ',' && e.KeyChar != '\b')

{

e.Handled = true;

}

}

/// <summary>

/// Ограничение на доступность кнопки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (accuracy.Text != "" && argumentX.Text != "") calc.Enabled = true;

else calc.Enabled = false;

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| **Проверяемые требования** | **Сообщения программы и вводимые значения** | **Ожидаемые результаты** | **Фактические результаты** |
| Способность программы обеспечить ввод исходных данных. | Ввести в поля ввода аргумент Х и точность e (в каждое поле вещественное число без знака с разделителем запятая) | В поле ввода можно ввести целое или вещественное число без знака, можно стирать символы в поле ввода, используя клавишу BackSpace. | Соответствуют ожиданиям. |
| Способность программы обеспечить контроль вводимых данных. | В поле ввода попробовать вводить буквы, специальные знаки, запятую, точку, знак минус. | Допустимыми являются только цифры, точка и запятая.  Недопустимые символы должны игнорироваться. | Соответствуют ожиданиям. |
| Способность программы обеспечить контроль пустых полей ввода. | Оставить одно или несколько полей ввода пустыми и нажать на кнопку «Вычислить». | Кнопка должна быть заблокированной | Кнопка заблокирована  Приложение не закрывается, пользователь имеет возможность продолжить работу (см. Рисунок 1) |
| Способность программы обеспечить контроль вводимых данных на вхождение в диапазон, указанный в условии | В поле ввода ввести данные, не соответствующие диапазону. | Должно быть выведено сообщение об ошибке | Сообщение об ошибке выводится.  Приложение не закрывается, пользователь имеет возможность продолжить работу (см. Рисунок 2, 3) |
| Проверить способность программы вычислять сумму ряда в зависимости от аргумента x и точности e | Ввести значения аргумента Х и точности е в поля ввода и нажать на кнопку «Вычислить». *Входные данные 1:*  Аргумент Х: 5  Точность е: 0,001  *Входные данные 2:*  Аргумент Х: 3  Точность е: 0,000001 | *Результат 1:* Сумма ряда = 1,3734008  *Результат 2:* Сумма ряда = 1,2490457 | Результаты соответствуют контрольным расчётам.  (см. Рисунок 4, 5) |

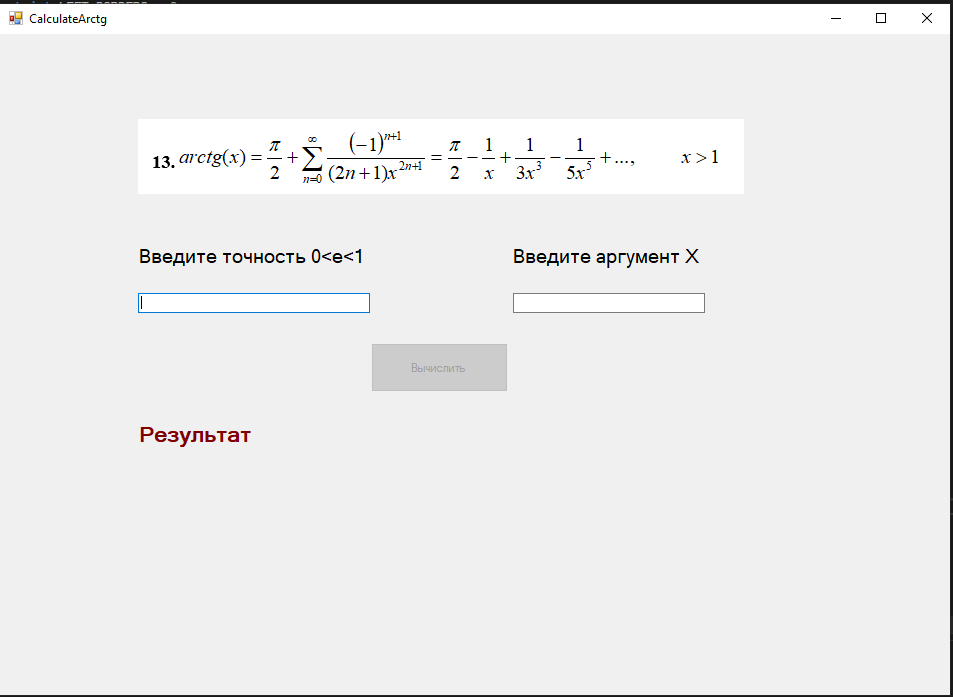


Рисунок 1.

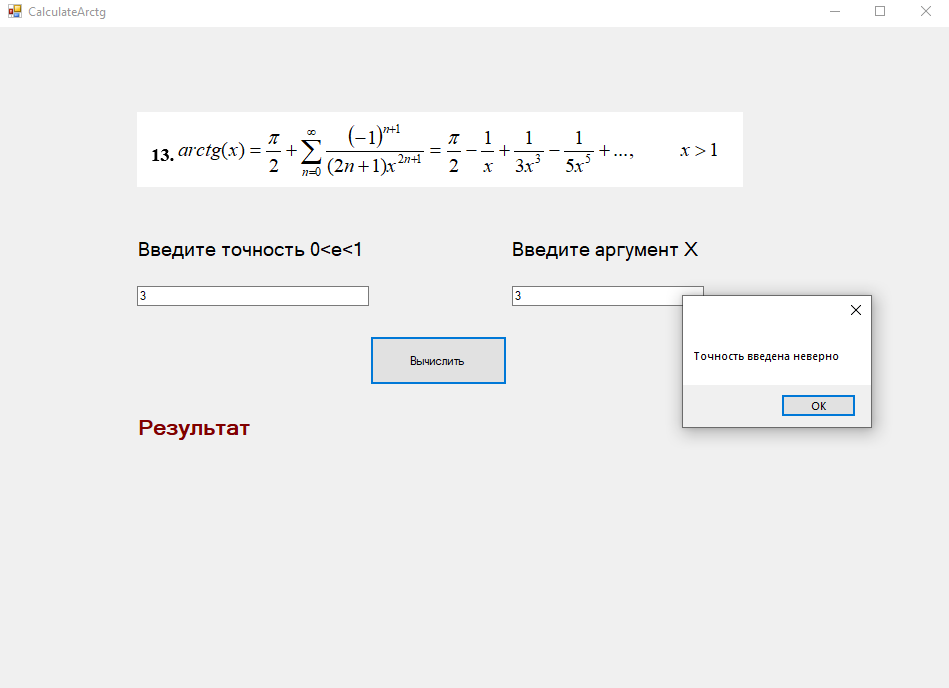


Рисунок 2.

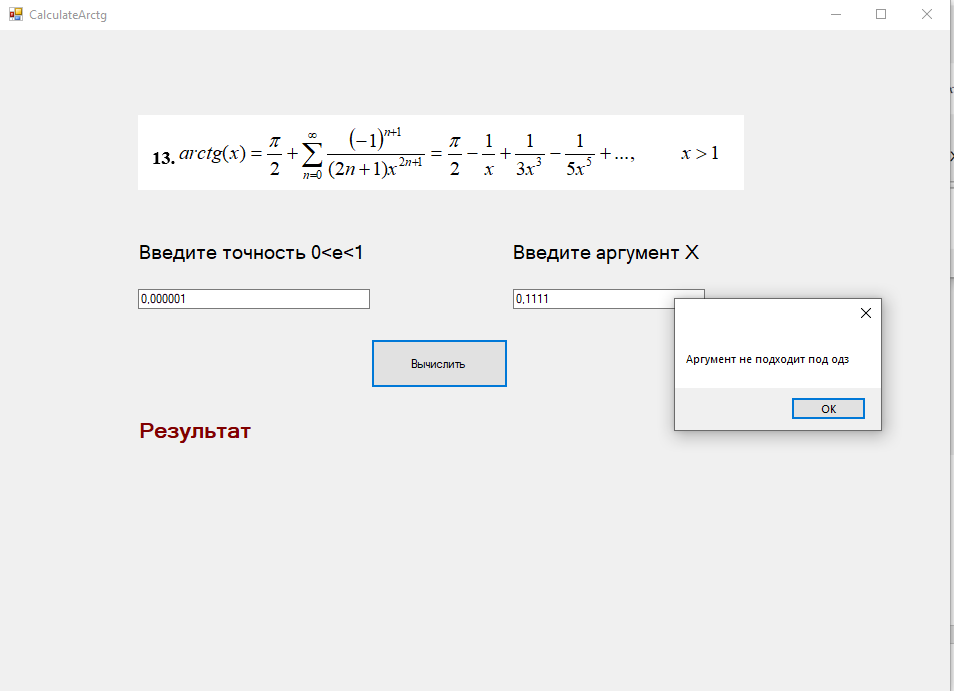


Рисунок 3.

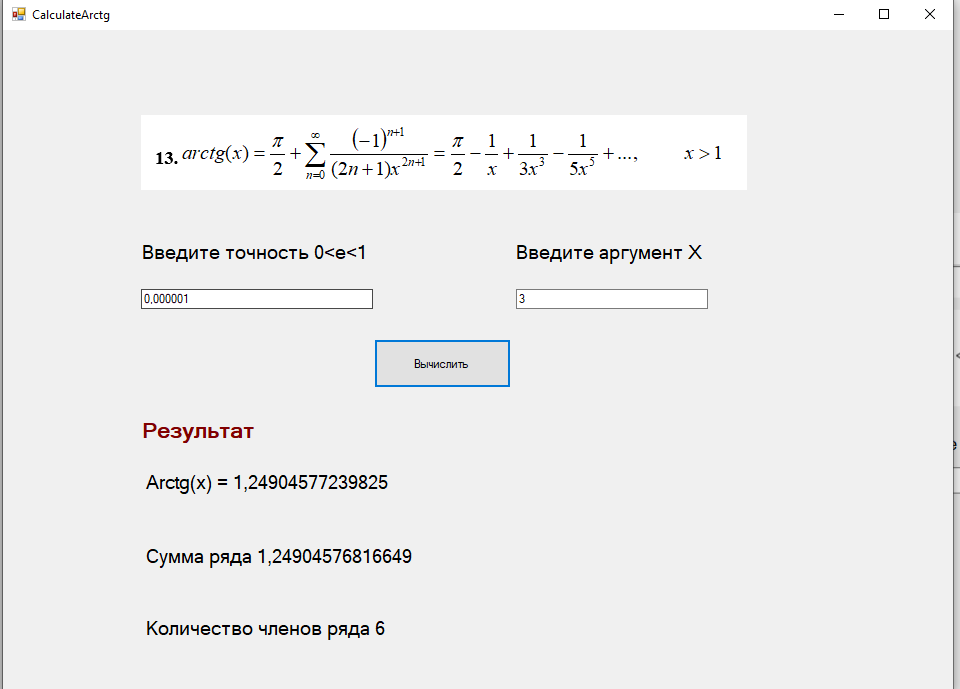


Рисунок 4.

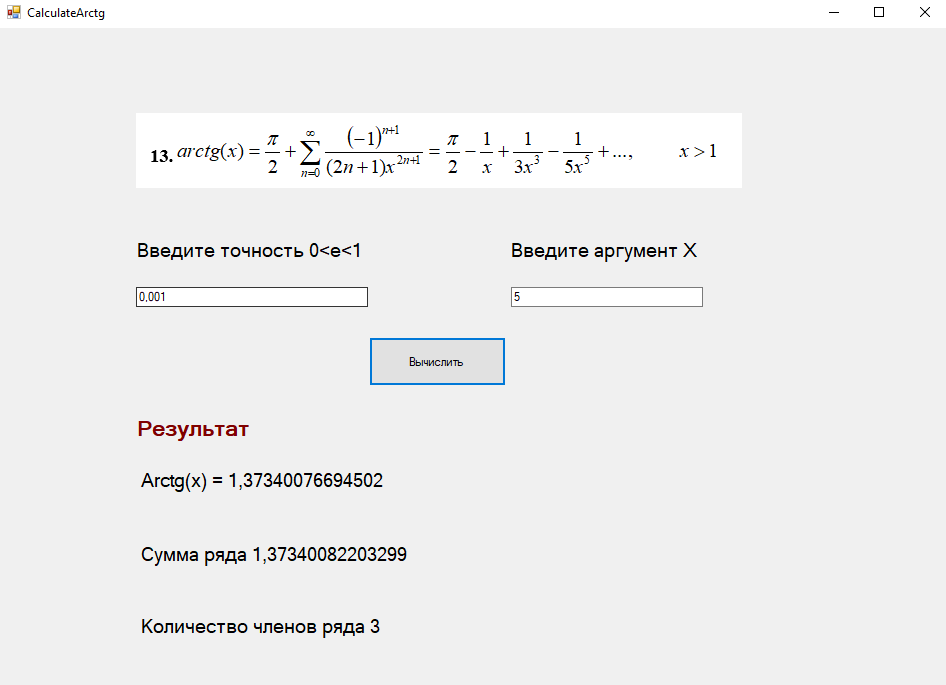


Рисунок 5.

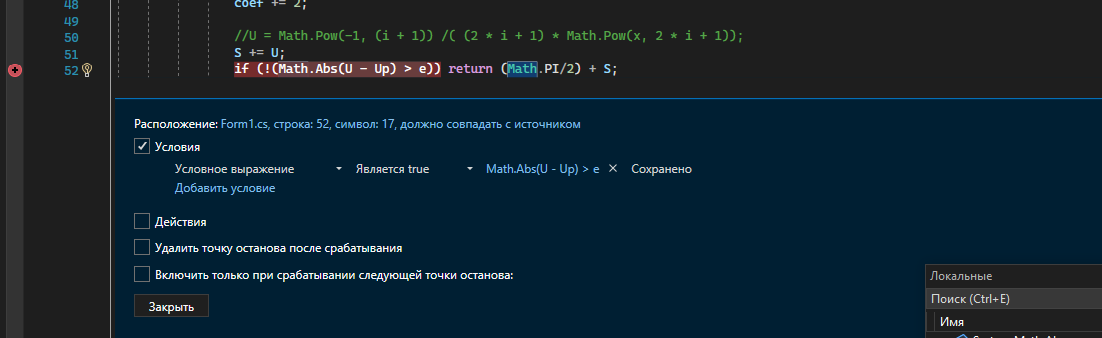
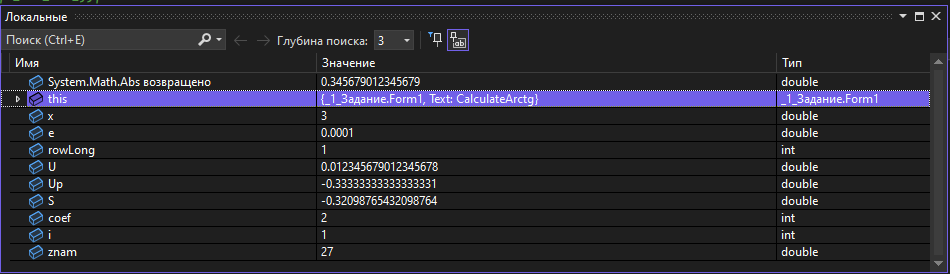


Рисунок 6. Установка параметра точки останова

Рисунок 7. Локальные переменные в конце итераций

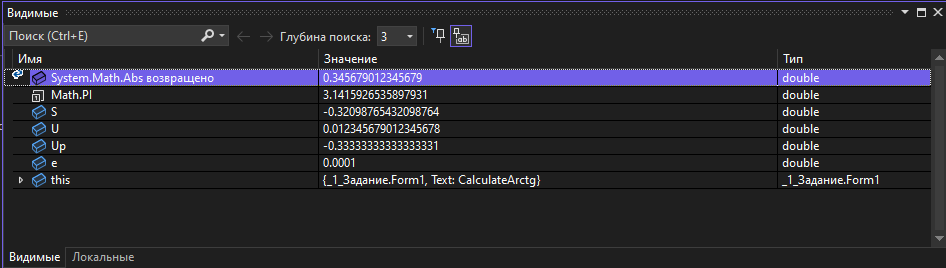


Рисунок 8. Просмотр видимых переменных

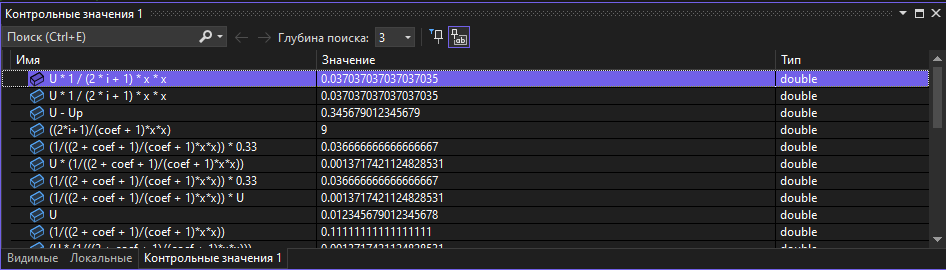


Рисунок 9. Просмотр контрольных значений

