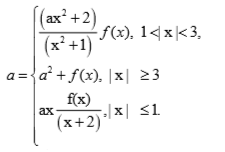
РАЗВЕТВЛЯЮЩИЕСЯ АЛГОРИТМЫ

**Цель лабораторной работы**: научиться пользоваться элементами управления для организации переключений (RadioButton). Написать и отладить программу разветвляющегося алгоритма.

**Задание:**



**Теория:**

Логические переменные и операции над ними

Переменные логического типа описываются посредством служебного слова bool. Они могут принимать только два значения – False (ложь) и True (истина). Результат False (ложь) и True (истина) возникает при использовании операций сравнения > (больше), < (меньше), != (не равно), >= (больше или равно), <= (меньше или равно), == (равно). Описываются логические переменные следующим образом:

bool b;

В языке C# имеются логические операции, применяемые к переменным логического типа. Это операции логического отрицания (!), логическое И (&&) и логическое ИЛИ (||). Операция логического отрицания является унарной операцией. Результат операции ! есть False, если операнд истинен, и True, если операнд имеет значение «ложь». Так,

!True → False (неправда есть ложь),

!False → True (не ложь есть правда).

Результат операции логическое И (&&) есть истина, только если оба ее операнда истинны, и ложь во всех других случаях. Результат операции логическое ИЛИ (||) есть истина, если какой-либо из ее операндов истинен, и ложен только тогда, когда оба операнда ложны.

Условные операторы

Операторы ветвления позволяют изменить порядок выполнения операторов в программе. К операторам ветвления относятся условный оператор if и оператор выбора switch.

Условный оператор if используется для разветвления процесса обработки данных на два направления. Он может иметь одну из форм: сокращенную или полную.

Форма сокращенного оператора if:

if (B) S;

где В – логическое или арифметическое выражение, истинность которого проверяется; S – оператор.

При выполнении сокращенной формы оператора if сначала вычисляется выражение B, затем проводится анализ его результата: если B истинно, то выполняется оператор S; если B ложно, то оператор S пропускается. Таким образом, с помощью сокращенной формы оператора if можно либо выполнить оператор S, либо пропустить его.

Форма полного оператора if:

if (B) S1; else S2;

где B – логическое или арифметическое выражение, истинность которого проверяется; S1, S2 – операторы.

При выполнении полной формы оператора if сначала вычисляется выражение B, затем анализируется его результат: если B истинно, то выполняется оператор S1, а оператор S2 пропускается; если B ложно, то выполняется оператор S2, а S1 – пропускается. Таким образом, с помощью полной формы оператора if можно выбрать одно из двух альтернативных действий процесса обработки данных.

Оператор выбора switch предназначен для разветвления процесса вычислений по нескольким направлениям. Формат оператора:

switch (<выражение>)  {

case <константное\_выражение\_1>:     [<оператор 1>];    <оператор перехода>;    case <константное\_выражение\_2>:     [<оператор 2>];    <оператор перехода>;     ...    case <константное\_выражение\_n>:     [<оператор n>];    <оператор перехода>;    [default:    <оператор>;]  }

Если ни одно выражение case не совпадает со значением оператора switch, управление передается операторам, следующим за необязательной подписью default. Если подписи default нет, то управление передается за пределы оператора switch.

Кнопки-переключатели

При создании программ в Visual Studio для организации разветвлений часто используются элементы управления в виде кнопок-переключателей (RadioButton). Состояние такой кнопки (включено–выключено) визуально отражается на форме, а в программе можно узнать его с помощью свойства Checked: если кнопка включена, это свойство будет содержать True, в противном случае False. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются. Группируются радиокнопки с помощью какого-либо контейнера – часто это бывает элемент GroupBox. Радиокнопки, размещенные в разных контейнерах, образуют независимые группы.

**Код программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace PP\_LR3

{

public partial class Form1 : Form

{

double f\_x;

public Form1()

{

InitializeComponent();

}

private void Calculate(object sender, EventArgs e)

{

try

{

double a = Convert.ToDouble(textBox1.Text.Replace('.', ','));

double x = Convert.ToDouble(textBox2.Text.Replace('.', ','));

double alpha = Convert.ToDouble(textBox3.Text.Replace('.', ','));

if (radioButton1.Checked)

f\_x = Math.Sinh(x);

if (radioButton2.Checked)

f\_x = Math.Pow(x, 2);

if (radioButton3.Checked)

f\_x = Math.Pow(Math.E, x);

if ((Math.Abs(x)) > 1 && (Math.Abs(x) < 3))

{

alpha = ((a \* Math.Pow(x, 2) + 2) / (Math.Pow(x, 2) + 1)) \* f\_x;

}

if (Math.Abs(x) >= x)

{

alpha = Math.Pow(alpha, 2) + f\_x;

}

if (Math.Abs(x) <= 1)

{

alpha = (a \* x) \* (f\_x / (x + 2));

}

listBox1.Items.Clear();

listBox1.Items.Add("При:");

listBox1.Items.Add("A = " + textBox1.Text);

listBox1.Items.Add("X = " + textBox2.Text);

listBox1.Items.Add("Alpha = " + textBox3.Text);

listBox1.Items.Add("Результат = " + alpha);

}

catch

{

MessageBox.Show("Ошибка! Введены неверные значения!");

}

}

private void Clear(object sender, EventArgs e)

{

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

listBox1.Items.Clear();

}

private void Exit(object sender, EventArgs e)

{

Application.Exit();

}

}

}

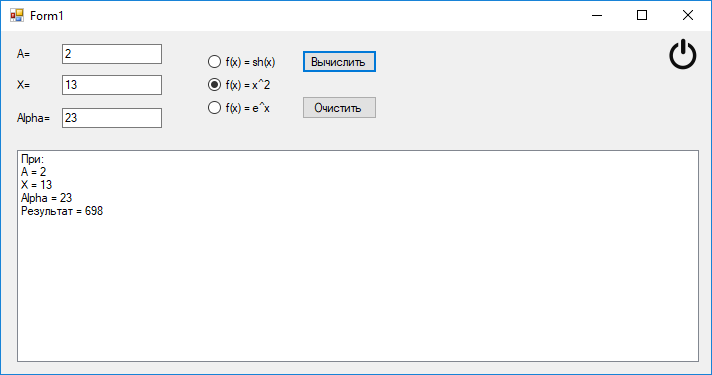


Рис.1 Работа программы

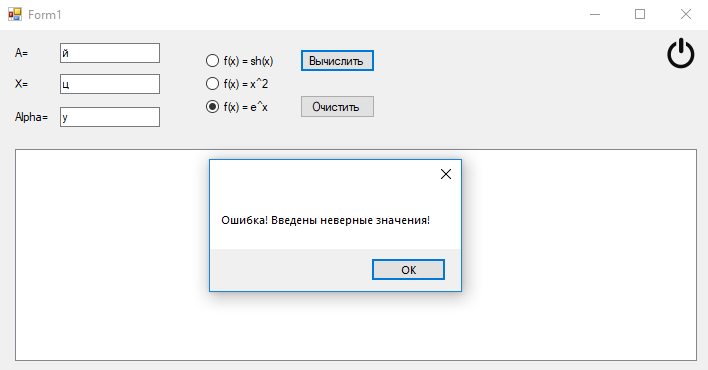


Рис.2 Введение неверных значений

**Вывод:**

Научились пользоваться элементами управления для организации переключений (RadioButton). Написали и отладили программу по заданным вариантам.