**ЛАБОРАТОРНАЯ РАБОТА №2. ПРОГРАММИРОВАНИЕ РАЗВЕТВЛЯЮЩИХСЯ АЛГОРИТМОВ**

Цель лабораторной работы: научиться пользоваться простейшими компонентами организации переключений (RadioButton). Написать и отладить программу разветвляющегося алгоритма.

2.1. 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 > меньше, < больше, != не равно, >= меньше или равно, <= больше или равно, == равно. Описываются логические переменные можно так:

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.

2.2. 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: простой или составной.

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.

ПРАКТИЧЕСКАЯ РАБОТА

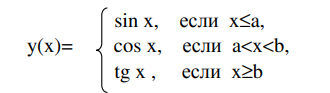
Форма полного оператора if:

if (B) S1; else S2;

где B - логическое или арифметическое выражение, истинность которого проверяется; S1, S2 - оператор: простой или составной.

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.

Пример. Вычислим значение функции



Указанное выражение может быть запрограммировано в виде

if (x<=a) y = Math.Sin(x);

if ((x>a) && (x<b)) y = Math.Cos(x);

if (x>=b) y = Math.Sin(x)/Math.Cos(x);

или

if (x <= a) y = Math.Sin(x);

else if (x < b) y = Math.Cos(x);

else y = Math.Sin(x) / Math.Cos(x);

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>; ]

}

Замечание. Выражение, записанное в квадратных скобках, является необязательным элементом в операторе switch. Если оно отсутствует, то может отсутствовать и оператор перехода.

Выражение, стоящее за ключевым словом switch, должно иметь арифметический, символьный, строковый тип или тип указатель. Все константные выражения должны иметь разные значения, но их тип должен совпадать с типом выражения, стоящим после switch или приводиться к нему. Ключевое слово case и расположенное после него константное выражение называют также меткой case.

Выполнение оператора начинается с вычисления выражения, расположенного за ключевым словом switch. Полученный результат сравнивается с меткой case. Если результат выражения соответствует метке case, то выполняется оператор, стоящий после этой метки, за которым обязательно должен следовать оператор перехода: break, goto и т.д. При использовании оператора break происходит выход из switch и управление передается оператору, следующему за switch. Если же используется оператор goto, то управление передается оператору, помеченному меткой, стоящей после goto. 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.

Пример использования оператора switch:

int caseSwitch = 1;

switch (caseSwitch)

{

case 1:

Console.WriteLine("Case 1");

break;

case 2:

Console.WriteLine("Case 2");

break;

default:

Console.WriteLine("Default case");

break;

}

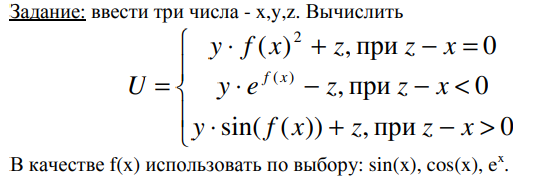
2.3. Кнопки-переключатели RadioButton

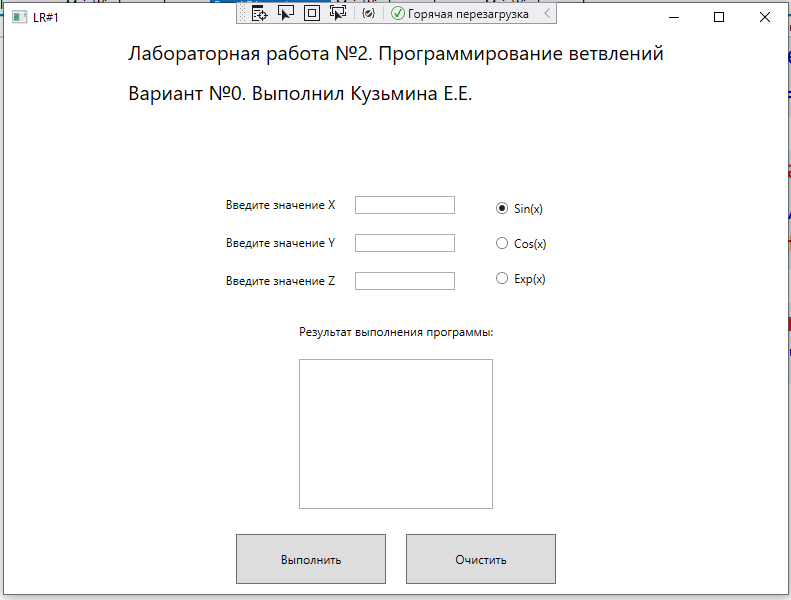
При создании программ в Visual Studio для организации разветвлений часто используются компоненты в виде кнопок-переключателей. Состояние такой кнопки (включено - выключено) визуально отражается на форме. Если пользователь выбирает один из вариантов переключателя в группе, все остальные автоматически отключаются.

Группу составляют все элементы управления RadioButton в заданном контейнере, таком как Form. Чтобы создать на одной форме несколько групп, поместите каждую группу в собственный контейнер, такой как элемент управления GroupBox или Panel. На форме (рис.3.1) представлены кнопки-переключатели RadioButton в контейнере GroupBox.

В программу передается номер включенной кнопки (0,1,2,..), который анализируется с помощью оператора switch.

2.4. Пример написания программы





<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="2\*"/>

<RowDefinition Height="3\*"/>

<RowDefinition Height="1\*"/>

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Orientation="Horizontal"

HorizontalAlignment="Center">

<StackPanel Orientation="Vertical"

VerticalAlignment="Center">

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Center">

<TextBlock Text="Введите значение X"

Margin="10"/>

<TextBox Name="txtX" Margin="10" Width="100"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Center">

<TextBlock Text="Введите значение Y"

Margin="10"/>

<TextBox Name="txtY" Margin="10" Width="100"/>

</StackPanel>

<StackPanel Orientation="Horizontal"

HorizontalAlignment="Center">

<TextBlock Text="Введите значение Z"

Margin="10"/>

<TextBox Name="txtZ" Margin="10" Width="100"/>

</StackPanel>

</StackPanel>

<StackPanel Orientation="Vertical"

VerticalAlignment="Center"

Margin="20">

<RadioButton GroupName="Function"

Name="rbtSin"

Content="Sin(x)"

IsChecked="True"

Margin="10"/>

<RadioButton GroupName="Function"

Name="rbtCos"

Content="Cos(x)"

Margin="10"/>

<RadioButton GroupName="Function"

Name="rbtExp"

Content="Exp(x)"

Margin="10"/>

</StackPanel>

</StackPanel>

<StackPanel Grid.Row="1" Orientation="Vertical"

HorizontalAlignment="Center">

<TextBlock Text="Результат выполнения программы:"

Margin="10"/>

<ListBox Name="lstResult" Margin="10"

Height="150"/>

</StackPanel>

<DockPanel Grid.Row="2"

HorizontalAlignment="Center">

<Button Name="btnCalc" Height="50"

Width="150"

Content="Выполнить" Margin="10"

Click="btnCalc\_Click"/>

<Button Name="btnClear" Height="50"

Width="150"

Content="Очистить" Margin="10"

Click="btnClear\_Click"/>

</DockPanel>

</Grid>

2.4.2. Создание обработчиков событий btn\_Calc и btn\_Clear

Обработчики событий создаются аналогично тому, как и в предыдущих лабораторных работах. ПРИМЕРНЫЙ текст обработчика события нажатия на кнопку ПУСК приведен ниже.

/// <summary>

/// ввод исходных данных

/// </summary>

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

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

private void btnCalc\_Click(object sender, RoutedEventArgs e)

{

lstResult.Items.Clear();

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

double y = double.Parse(txtY.Text);

double z = double.Parse(txtZ.Text);

double u;

lstResult.Items.Add("Лаб.раб.№2. Выполнил Кузьмина Е.Е.");

lstResult.Items.Add($"X={x}");

lstResult.Items.Add($"Y={y}");

lstResult.Items.Add($"Z={z}");

int n = 0;

if (rbtCos.IsChecked == true) n = 1;

else if (rbtExp.IsChecked == true) n = 2;

switch (n)

{

case 0:

if ((z - x) == 0) u = y \* Math.Sin(x) \* Math.Sin(x) + z;

else if ((z - x) < 0) u = y \* Math.Exp(Math.Sin(x)) - z;

else u = y \* Math.Sin(Math.Sin(x)) + z;

lstResult.Items.Add($"Результат U={Math.Round(u, 3)}");

break;

case 1:

if ((z - x) == 0) u = y \* Math.Cos(x) \* Math.Cos(x) + z;

else if ((z - x) < 0) u = y \* Math.Exp(Math.Cos(x)) - z;

else u = y \* Math.Sin(Math.Cos(x)) + z;

lstResult.Items.Add($"Результат U={Math.Round(u, 3)}");

break;

case 2:

if ((z - x) == 0) u = y \* Math.Exp(x) \* Math.Exp(x) + z;

else if ((z - x) < 0) u = y \* Math.Exp(Math.Exp(x)) - z;

else u = y \* Math.Sin(Math.Exp(x)) + z;

lstResult.Items.Add($"Результат U={Math.Round(u, 3)}");

break;

default:

lstResult.Items.Add("Решение не найдено");

break;

}

}

/// <summary>

/// Очистить поля

/// </summary>

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

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

private void btnClear\_Click(object sender, RoutedEventArgs e)

{

txtX.Clear();

txtY.Clear();

txtZ.Clear();

lstResult.Items.Clear();

}

2.5. Выполнение индивидуального задания

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве f(x) использовать по выбору: sh(x), x2, ex.

Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

