**Практическая работа №3. Программирование разветвляющихся алгоритмов**

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

***3.1. 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.

***3.2. 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.

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

x, если x≤a,

F(x)= x+2, если a<x<b,

ex , если x≥b .

Указанное выражение может быть запрограммировано в виде

**if (x<=a) y = x;if ((x>a) && (x<b)) y = x+2;if (x>=b) y = Math.Exp(x);**

или

**if (x <= a) y = x;**

**else if (x < b) y = x+2;**

**else y = Math.Exp(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;**

}

***3.3. 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**.

***3.4. Пример написания программы***

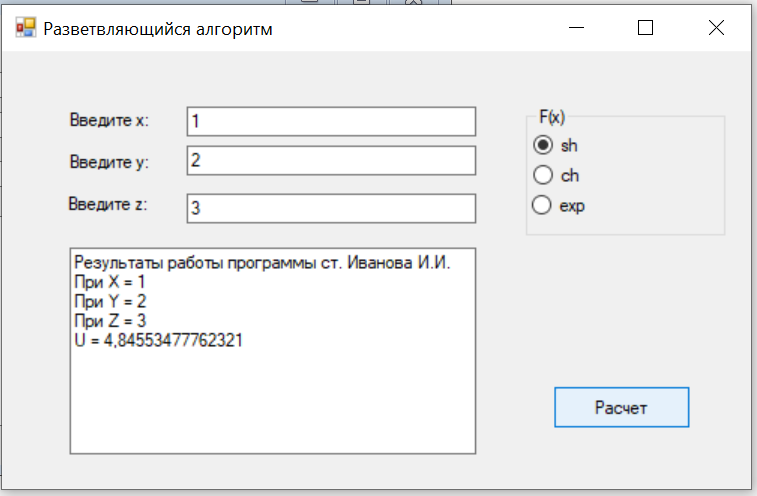
Задание: ввести три числа - x,y,z. Вычислить



В качестве f(x) использовать по выбору: sh(x), ch(x), ex.

***3.4.1. Создание формы***

Создайте форму, в соответсвии с рис. 3.1.



*Рис 3.1. Окно практической работы*

Выберите в панели элементов из контейнеров **GroupBox** и поместите его в нужное место формы. На форме появится окаймленный линией чистый прямоугольник с заголовком **GroupBox1**. Замените заголовок (**Text**) на F(x). Далее, как показано на рисунке, разместите в данном контейнере три радиокнопки (**RadioButton**). Для первой из них установите свойство **Checked** в значение **True**.

Далее разместите на форме элементы **Label**, **TextBox** и **Button**. Поле для вывода результатов также является элементом **TextBox** с установленным в **True** свойством **Multiline**.

***3.4.2. Создание обработчиков событий FormСreate и Botton1Сlick***

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

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

{

// Получение исходных данных из TextBox

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

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

double z = Convert.ToDouble(textBox3.Text);

// Ввод исходных данных в окно результатов

textBox4.Text = "Результаты работы программы ст. Иванова И.И. " + Environment.NewLine;

textBox4.Text += "При X = " + textBox1.Text + Environment.NewLine;

textBox4.Text += "При Y = " + textBox2.Text + Environment.NewLine;

textBox4.Text += "При Z = " + textBox3.Text + Environment.NewLine;

// Определение номера выбранной функции

int n = 0;

if (radioButton2.Checked) n = 1;

else if (radioButton3.Checked) n = 2;

// Вычисление U

double u;

switch (n)

{

case 0:

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

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

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

textBox4.Text += "U = " + Convert.ToString(u) + Environment.NewLine;

break;

case 1:

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

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

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

textBox4.Text += "U = " + Convert.ToString(u) + Environment.NewLine;

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;

textBox4.Text += "U = " + Convert.ToString(u) + Environment.NewLine;

break;

default:

textBox4.Text += "Решение не найдено" + Environment.NewLine;

break;

}

}

Запустите программу и убедитесь в том, что все ветви алгоритма выполняются правильно.

***3.5. Выполнение индивидуального задания***

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

1. y= 9.y=

2.y= 10.y=

3.y= 11.y=

4.y= 12.y=

5.y= 13y=

6.y= 14.y=

7.y= 15.y=

8.y= 16.y=