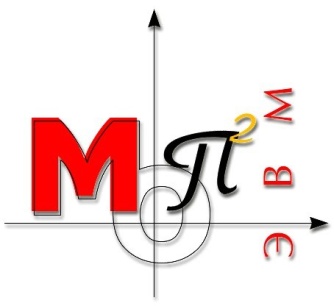
Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждения высшего образования   
«Южный федеральный университет»

Инженерно-технологическая академия

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**

Лабораторная работа №5

по дисциплине

"ООП"

на тему

" Классы, объекты, наследование в С#"

**Выполнил:**

студент группы КТбо2-6

Гришак В.С.

**Проверил:**

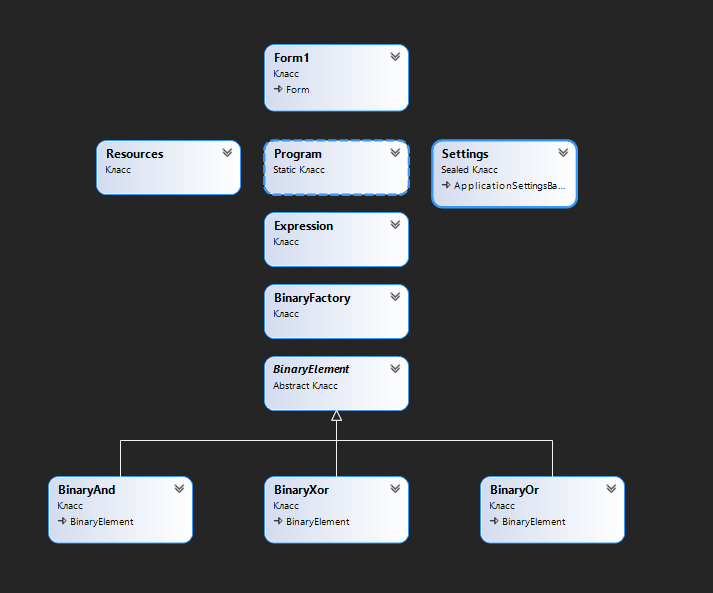
Преподаватель  
Тарасов С.А.

**ВАРИАНТ №10**

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Описать класс Element (элемент логической схемы) с двумя входами и одним выходом и полем, хранящим название элемента. Определить функцию, которая преобразует входные двоичные значения в выходное. На его основе реализовать классы AND и OR - двоичные вентили, которые реализуют логическое умножение и сложение соответственно. В дополнительном классе Sсheme создать массив элементов (до 10) и обеспечить подачу двоичных сигналов на их входы с выводом выходных значений. Входные сигналы хранятся в файле.

**ДИАГРАММА КЛАССОВ**

****

**ЛИСТИНГ КОДА**

**Expression.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace BinaryCalculator

{

class Expression

{

public string Execute(string strExpression)

{

List<string> listExpression = ReadExpression(strExpression);

if (listExpression.Count % 2 == 0)

{

throw new Exception("Неккоректное выражение!");

}

Queue<string> postfixNotation = PostfixNotation(listExpression);

Stack<string> expression = new Stack<string>();

BinaryFactory binaryFactory = new BinaryFactory();

while (postfixNotation.Count != 0)

{

if (binaryFactory.CheckOperation(postfixNotation.Peek()))

{

string firstOperand, secondOperand;

if (expression.Count != 0)

{

firstOperand = expression.Pop();

if (expression.Count != 0)

{

secondOperand = expression.Pop();

}

else

{

throw new Exception("Некорректное выражение");

}

}

else

{

throw new Exception("Некорректное выражение");

}

expression.Push(binaryFactory.CreateElement(postfixNotation.Dequeue()).

Operation(firstOperand, secondOperand));

}

else

{

expression.Push(postfixNotation.Dequeue());

}

}

if (expression.Count != 1)

{

throw new Exception("Некорректное выражение!");

}

else

{

return RemoveZeros(expression.Pop());

}

}

private string StrExpressionCorrect(List<string> listExpression)

{

StringBuilder str = new StringBuilder();

foreach (string elem in listExpression)

{

str.Append(elem + " ");

}

str.Remove(str.Length - 1, 1);

return str.ToString();

}

private List<string> ReadExpression(string expression)

{

List<string> listExpression = new List<string>();

string operators = "&|^()";

for (int i = 0; i < expression.Length; i++)

{

if (expression[i] == ' ')

{

continue;

}

else if (expression[i] == '1' || expression[i] == '0')

{

StringBuilder str = new StringBuilder();

while (expression[i] == '1' || expression[i] == '0')

{

str.Append(expression[i]);

i++;

if (i == expression.Length)

{

break;

}

}

i--;

listExpression.Add(str.ToString());

}

else if (operators.Contains(expression[i].ToString()))

{

listExpression.Add(expression[i].ToString());

}

else

{

throw new Exception("Некорректный символ!");

}

}

return listExpression;

}

private string RemoveZeros(string str)

{

while (str.Length - 1 != 0)

{

if (str[0] == '0')

{

str = str.Remove(0, 1);

}

else

{

break;

}

}

return str;

}

private Queue<string> PostfixNotation(List<string> listExpression)

{

Stack<string> stackOperators = new Stack<string>();

Queue<string> postfixNotation = new Queue<string>();

foreach (string str in listExpression)

{

if (str == "&")

{

if (stackOperators.Count == 0)

{

stackOperators.Push(str);

}

else if (stackOperators.Peek() == "(" || stackOperators.Peek() != "&")

{

stackOperators.Push(str);

}

else

{

string pop = stackOperators.Peek();

while (pop != "(" && pop != "|" && pop != "^")

{

postfixNotation.Enqueue(stackOperators.Pop());

if (stackOperators.Count != 0)

{

pop = stackOperators.Peek();

}

else

{

break;

}

}

stackOperators.Push(str);

}

}

else if (str == "^" || str == "|")

{

if (stackOperators.Count == 0)

{

stackOperators.Push(str);

}

else if (stackOperators.Peek() == "(")

{

stackOperators.Push(str);

}

else

{

while (stackOperators.Peek() != "(")

{

postfixNotation.Enqueue(stackOperators.Pop());

if (stackOperators.Count==0)

{

break;

}

}

stackOperators.Push(str);

}

}

else if (str == "(")

{

stackOperators.Push(str);

}

else if (str == ")")

{

if (stackOperators.Count != 0)

{

while (stackOperators.Peek() != "(")

{

postfixNotation.Enqueue(stackOperators.Pop());

if (stackOperators.Count == 0)

{

throw new Exception("Отсутствует открывающая скобка!");

}

}

stackOperators.Pop();

}

else

{

throw new Exception("Отсутствует открывающая скобка!");

}

}

else

{

postfixNotation.Enqueue(str);

}

}

while (stackOperators.Count != 0)

{

postfixNotation.Enqueue(stackOperators.Pop());

}

return postfixNotation;

}

}

}

**Form1.cpp**

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 BinaryCalculator

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

try

{

textBox2.Text = (new Expression()).Execute(textBox1.Text);

}

catch(Exception error)

{

MessageBox.Show(error.Message);

}

}

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

{

textBox1.Text = "";

textBox2.Text = "";

}

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

{

if (textBox1.SelectedText.Length != 0)

{

textBox1.SelectedText = "";

}

else

{

if (textBox1.SelectionStart != 0)

{

int pos = textBox1.SelectionStart;

textBox1.Text = textBox1.Text.Remove(pos - 1, 1);

textBox1.SelectionStart = pos - 1;

}

}

}

private void buttonNum0\_Click(object sender, EventArgs e)

{

textBox1.SelectedText = "0";

}

private void buttonNum1\_Click(object sender, EventArgs e)

{

textBox1.SelectedText = "1";

}

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

{

textBox1.SelectedText = "&";

}

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

{

textBox1.SelectedText = "|";

}

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

{

textBox1.SelectedText = "^";

}

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

{

textBox1.SelectedText = "(";

}

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

{

textBox1.SelectedText = ")";

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyValue == (char)Keys.Enter)

{

buttonCalculate\_Click(buttonCalculate, null);

}

}

}

}