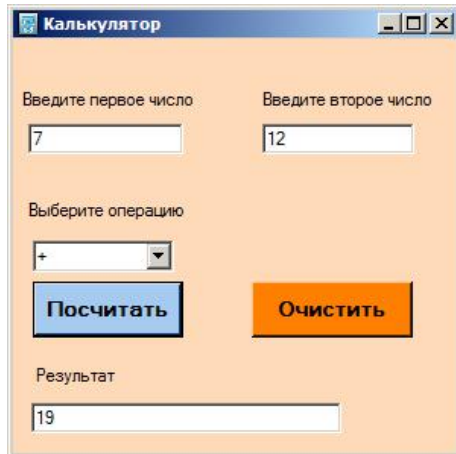


Лабораторная работа № 7-8

Тестирование калькулятора на C# в Windows Forms

Цель: научиться создавать различные вариации арифметического калькулятора на языке программирования C# и тестировать их работу, сравнивая по различным критериям.

Задание 1. Создание простого калькулятора в среде Windows Forms Application



```
private void button1_Click(object sender, EventArgs e)
{
    double a = Convert.ToDouble(textBox1.Text);
    double b = Convert.ToDouble(textBox2.Text);

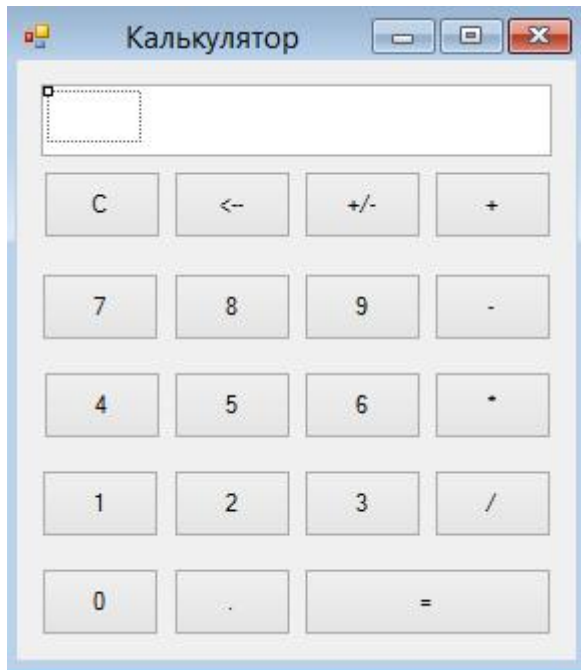
    switch (comboBox1.Text)
    {
        case "+":
            textBox3.Text = Convert.ToString(a + b);
            break;
        case "-":
            textBox3.Text = Convert.ToString(a - b);
            break;
        case "*":
            textBox3.Text = Convert.ToString(a * b);
            break;
        case "/":
            if (b == 0)
                MessageBox.Show("На ноль делить нельзя!", "Ошибка");
            textBox3.Text = Convert.ToString(a / b);
            break;
    }
}

//ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
    textBox2.Text = "";
    textBox3.Text = "";
    comboBox1.Text = "";
}
```

Задание 2. Создание второго калькулятора на C#

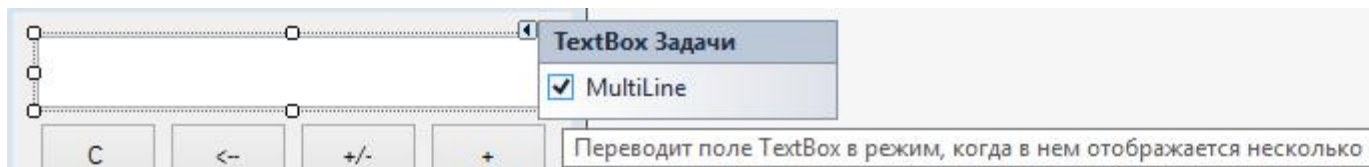
Сейчас мы создадим более усовершенствованный калькулятор Windows Forms.

Выглядеть у нас он будет вот так:

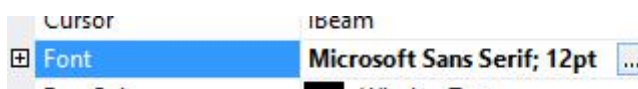


Здесь у нас 19 кнопок **Button**, 1 **Textbox** и ещё 1 пустой **Label** (на рисунке он выделен). Применение его будет описано ниже.

Итак, создаём такую или похожую форму. Мы увеличили ширину **TextBox**'а, используя **MultiLine**:



Также в Свойствах мы увеличили размер шрифта в **TextBox**'е и **Label**'е до 12 пт.

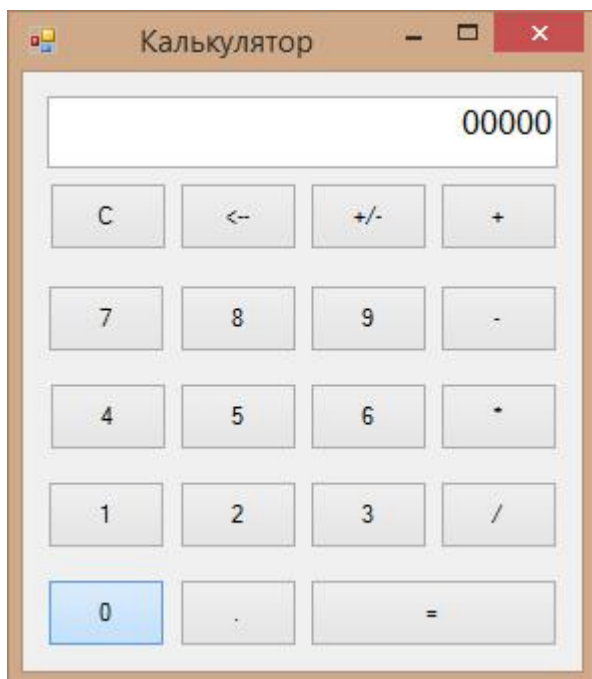


Теперь делаем так, чтобы при нажатии на цифровые кнопки, в **TextBox**'е появлялась соответствующая цифра.

Для этого дважды кликаем на кнопке “0” и в открывшемся коде пишем:

```
1 private void button17_Click(object sender, EventArgs e)
2     {
3         textBox1.Text = textBox1.Text + 0;
4     }
```

Проверяем, несколько раз нажав на кнопку “0” у нас в форме.



Работает. Делаем то же самое с остальными цифровыми кнопками:

```

private void button13_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 1;
}

private void button14_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 2;
}

private void button15_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 3;
}

private void button9_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 4;
}

private void button10_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 5;
}

private void button11_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 6;
}

private void button5_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 7;
}

private void button6_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 8;
}

private void button7_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + 9;
}

```

Таким же образом кликаем дважды на кнопку “.” в форме. Она будет использоваться для создания десятичной дроби. Пишем следующий код:

```

private void button18_Click(object sender, EventArgs e)
{
    textBox1.Text = textBox1.Text + ",";
}

```

Кнопки нажимаются, в **TextBox**’е отображаются нажатые цифры. Теперь надо научить программу производить с ними какие-либо операции. Как видно из формы, наш калькулятор сможет производить стандартные математические операции: сложение, вычитание, умножение и деление. Для начала мы создадим в самом начале программы несколько переменных, которые нам для этого понадобятся:

```

float a, b;
int count;
bool znak = true;

```

Первым двум переменным будут присваиваться значения, набранные пользователем в калькуляторе. В последствии с ними будут производиться нужные математические операции. Тип **float** – это тип с плавающей точкой, позволяющий работать с десятичными дробями, что нам, безусловно, нужно при наличии кнопки “.”.

Благодаря второй переменной мы будем давать программе указания, какую именно операцию производить с переменными, описанными выше. Здесь нам не нужна дробь, поэтому обойдёмся целочисленным типом **int**.

Последняя переменная **znak** нам понадобится для того, чтобы менять знаки у введённых чисел. Тип **bool** может иметь два значения – **true** и **false**. Мы представим, что если **znak** имеет значение **true** в программе, то это означает, что у числа знак +, если **false** – число отрицательное и перед собой имеет знак -. Изначально в калькуляторе вбиваются положительные числа, поэтому мы сразу присвоили переменной значение **true**.

Далее мы дважды нажимаем на кнопку “+”, обозначающую сложение, на форме и пишем следующий код:

```
private void button4_Click(object sender, EventArgs e)
{
    a = float.Parse(textBox1.Text);
    textBox1.Clear();
    count = 1;
    label1.Text = a.ToString() + "+";
    znak = true;
}
```

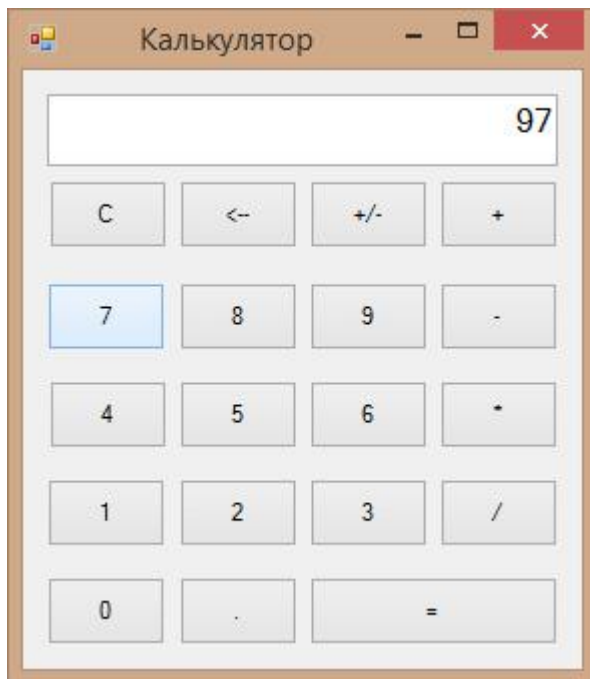
В строке 3 мы присваиваем первой переменной **a** то, что будет написано в **TextBox’e** (а именно число, которое введёт пользователь перед тем, как нажать кнопку “+”).

Затем **TextBox** очищается, число, введённое пользователем, в нём пропадает (но остаётся в переменной **a**)

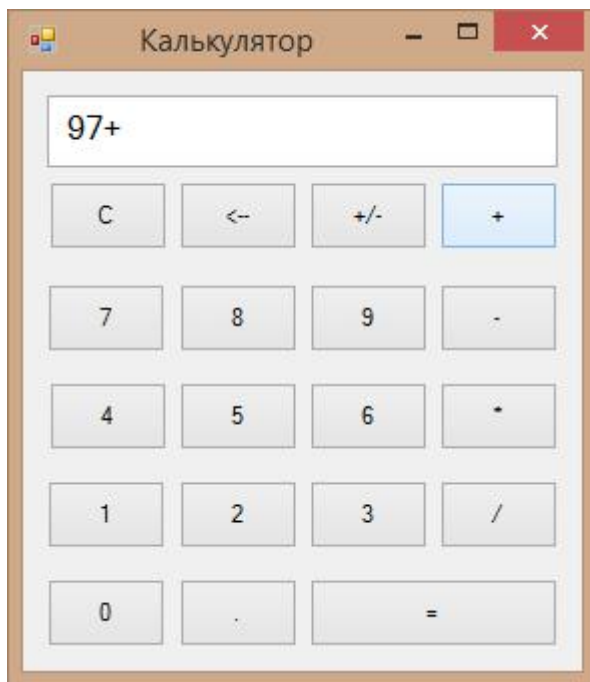
Переменной **count** присваивается число 1, которая потом укажет программе, что именно операцию сложения надо будет произвести с числами.

Затем в **Label** записывается число из переменной **a** (то самое, которое изначально ввёл пользователь) и знак плюса. Выглядеть в форме это будет так, как описано ниже.

Пользователь вводит какое-либо число:



Затем нажимает на кнопку “+” и после этого видит:



Кроме того, как бы не было странным с первого взгляда, мы присваиваем переменной **znak** значение **true**, хотя выше, в начале кода, мы и так присваивали это же значение. Подробнее данную переменную мы опишем ниже, но смысл в том, что мы присваиваем значение **true**, когда хотим сделать введенное число отрицательным, если оно положительное, а значение **false**, когда хотим сделать число положительным, если оно отрицательное. Изначально у нас вводятся положительные числа, сначала первое, потом второе. И если первое число мы сделаем отрицательным, то значение у **znak** перейдет в **false** и тогда получится, что второе слагаемое как бы отрицательное (на практике, просто чтобы поставить перед ним минус, придется нажать дважды на соответствующую кнопку, чтобы с **false** значение перешло в **true**, а затем обратно с **true** в **false**, и появился знак минуса)

Подобным образом заполняем код для кнопок “-“, “*” и “/”:

```
private void button8_Click(object sender, EventArgs e)
{
    a = float.Parse(textBox1.Text);
    textBox1.Clear();
    count = 2;
    label1.Text = a.ToString() + "-";
    znak = true;
}

private void button12_Click(object sender, EventArgs e)
{
    a = float.Parse(textBox1.Text);
    textBox1.Clear();
    count = 3;
    label1.Text = a.ToString() + "*";
    znak = true;
}

private void button16_Click(object sender, EventArgs e)
{
    a = float.Parse(textBox1.Text);
    textBox1.Clear();
    count = 4;
    label1.Text = a.ToString() + "/";
    znak = true;
}
```

Разница лишь в значении переменной **count** и в том, какой знак добавляется в **Label’e**.

Далее нам понадобится создать функцию, которая будет применять нужные нам математические операции к числам. Назовём её **calculate**. Но перед этим мы кликнем дважды на кнопку “=” на форме и в коде к ней мы запишем:

```
private void button19_Click(object sender, EventArgs e)
{
    calculate();
    label1.Text = "";
}
```

То есть, при нажатии пользователем на кнопку “=”, как раз выполнится наша функция подсчёта **calculate**, и, заодно, очистится **Label**, так как результат мы в будущем коде выведем в **TextBox**.

Теперь-таки создаём нашу функцию **calculate** и пишем следующий код:


```

private void calculate()
{
    switch(count)
    {
        case 1:
            b = a + float.Parse(textBox1.Text);
            textBox1.Text = b.ToString();
            break;
        case 2:
            b = a - float.Parse(textBox1.Text);
            textBox1.Text = b.ToString();
            break;
        case 3:
            b = a * float.Parse(textBox1.Text);
            textBox1.Text = b.ToString();
            break;
        case 4:
            b = a / float.Parse(textBox1.Text);
            textBox1.Text = b.ToString();
            break;

        default:
            break;
    }
}

```

Здесь мы используем конструкцию **switch-case**.

Switch – это оператор управления. Он может включать в себя несколько **case’ов**. **Case** – метки, от значения которых зависит, какие операции будут происходить.

Строка **switch(count)** означает, что именно от значения **count** будет зависеть, какое действие будет происходить в коде **switch’a**.

Итак, если **count=1** (в коде **case 1:**), то произойдёт следующее:

После того, как пользователь нажал “+”, он, естественно, должен ввести второе слагаемое, что он и делает по стандартному сценарию, а затем нажать кнопку “=” (и в ней, как мы помним, как раз выполнится наша функция).

Как только кнопка “=” будет нажата, программа сложит число из переменной **a** с тем вторым слагаемым, которое записал пользователь в **TextBox**, и запишет результат в переменную **b** (строка 6 кода функции). В строке 7 программа выведет в **TextBox** результат сложения – переменную **b**.

Оператор **break** (строка 8) завершает исполнение кода **switch** при выполнении кода метки **case 1**, так как больше нам в нём делать нечего.

Точно так же строится алгоритм при **case 2**, **case 3** и **case 4** с той разницей, что в них происходит не сложение, а вычитание, умножение и деление соответственно.

Оператор **default** срабатывает, если вдруг что-то пойдёт не по плану и **count** примет какое-либо иное значение, не описанное в **switch**. Тогда срабатывает лишь оператор **break**.

Львиная доля программы готова. Нам надо лишь написать код для трёх оставшихся нетронутыми до этого время кнопок.

Дважды жмём в форме на кнопку “C”. Она будет очищать все записи из **TextBox’a** и **Label’a**.

Код у неё элементарный:

```
private void button3_Click(object sender, EventArgs e)
{
    textBox1.Text = "";
    label1.Text = "";
}
```

На очереди у нас кнопка “←”. Она будет удалять последнюю цифру записанного в **TextBox’e** числа. Код:

```
private void button2_Click(object sender, EventArgs e)
{
    int lenght = textBox1.Text.Length - 1;
    string text = textBox1.Text;
    textBox1.Clear();
    for (int i = 0; i < lenght; i++)
    {
        textBox1.Text = textBox1.Text + text[i];
    }
}
```

Мы вводим новую переменную **lenght** целочисленного типа и записываем в неё количество символов в **TextBox’e** минус один символ.

Также мы вводим новую переменную **text**, в которую полностью заносим текст из **TextBox’a**. Затем мы очищаем **TextBox** и вводим цикл **for**, через который записываем в **TextBox** строку **text**, но уже на символ короче.

Например, в **TextBox’e** записано число *504523*

При нажатии на кнопку “←” в переменную **lenght** записывается число 5 (6 цифр – 1), в переменную **text** записывается строка “504523”, **TextBox** очищается, а затем в него по одному записываются символы из **text**, но в этот раз их будет не 6, а 5, то есть в **TextBox’e** появится число *50452*.

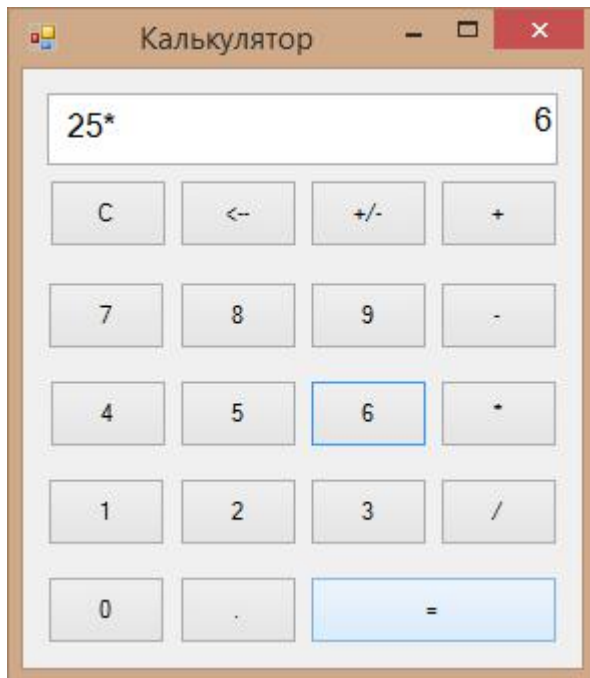
У нас остаётся последняя кнопка, которая отвечает за знак первого слагаемого. Переходим к её коду. Тут мы будем работать с переменной **znak**, которую описывали выше. Код выглядит вот так:

```
private void button1_Click(object sender, EventArgs e)
{
    if(znak==true)
    {
        textBox1.Text = "-" + textBox1.Text;
        znak = false;
    }
    else if (znak==false)
    {
        textBox1.Text=textBox1.Text.Replace("-", "");
        znak = true;
    }
}
```

Изначально, как мы помним, у переменной **znak** стоит значение **true**. Если мы нажмём на кнопку первый раз, то в **TextBox'е** перед числом появится знак “-“, а переменной **znak** будет присвоено значение **false**.

Если второй раз нажать на данную кнопку, то, так как **znak** у нас **false**, произойдёт второе условие. Здесь используется метод **Replace**, который заменяет какой-либо кусок строки на другой. В скобках после метода вначале пишется, что будет заменено в строке, а после запятой, то, на что заменять. В данном случае мы заменяем в **TextBox'е** минус на пустое значение.

Вот и всё, наш калькулятор Windows Forms готов! Можно его тестировать!



Код для вывода времени выполнения приложения:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

namespace Калькулятор2
{
    ссылка: 0
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        ссылка: 0
        static void Main()
        {
            Stopwatch st = new Stopwatch();
            st.Start();
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
            st.Stop();
            MessageBox.Show("Время выполнения: " + st.Elapsed.TotalSeconds);
        }
    }
}
```

Задание 3. Разработать собственный третий калькулятор по вариантам:

Вариант	Задание
1	Калькулятор, который осуществляет перевод числа в двоичную, восьмеричную и шестнадцатеричную систему счисления
2	Калькулятор, который осуществляет перевод числа из двоичной, восьмеричной и шестнадцатеричной в десятичную систему счисления.
3	Калькулятор, который вычисляет x^2 x^3 x^y $\sqrt[y]{x}$ $\sqrt[x]{y}$ 10^x
4	Калькулятор, который вычисляет Exp Mod log n! 1/x
5	Калькулятор, который вычисляет Sin, Cos, Tg, Ctg угла
6	Калькулятор, который вычисляет Arcsin, Arccos, Arctg, Arcctg чисел
7	Калькулятор, который вычисляет Sin и Cos угла: <input checked="" type="radio"/> Градусы <input type="radio"/> Радианы <input type="radio"/> Грады
8	Кредитный калькулятор (сумма, процентная ставка, срок, ежемесячный платеж)
9	Калькулятор, который вычисляет остаток от деления, квадратный корень, квадрат числа, факториал числа
10	Калькулятор, который вычисляет Mod Exp x^y $\sqrt[y]{x}$

Задание 4. Протестировать все 3 созданных калькулятора по следующей таблице:

Номер п/п	Вид тестирования	Результат проверки 1 калькулятора	Результат проверки 2 калькулятора	Результат проверки 3 калькулятора	Вывод
1	Ввод данных	Осуществляется вводом в текстовое поле с клавиатуры	Осуществляется нажатием мышью на соответствующие кнопки + вводом в текстовое поле с клавиатуры		
2	Реакция системы на введенные данные				
3	Точка-разделитель дробной части				
4	Способы ввода операции				
5	Арифметические операции				
6	Получение результата				
7	Удаление последнего символа				
8	Сброс				
9	Специальные тесты				
10	Деление на 0				

11	Завершение работы				
12	Отображение процесса операции				
13	Возможность вводить отрицательные числа				
14	Скорость работы калькулятора				
15	Сколько памяти занимает				

Задание 5. Оформить отчет.