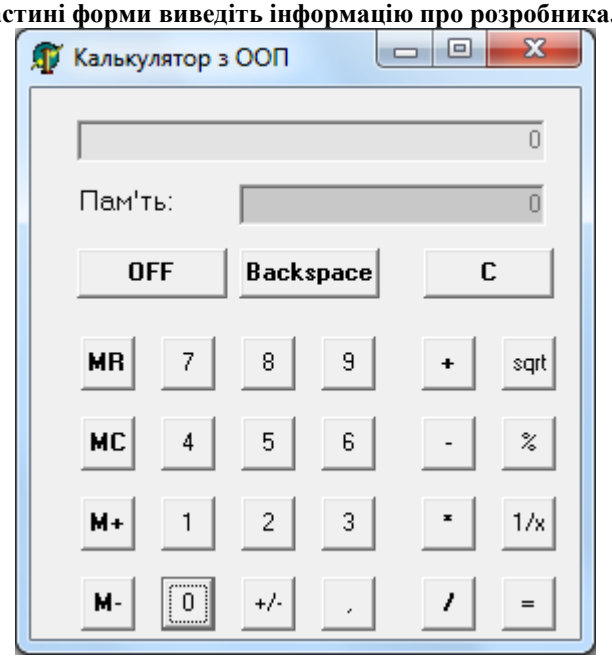
Лабораторна робота № 11

Тема. Організація взаємодії елементів керування у формах

Мета. Формування вмінь і навиків програмування алгоритмів обробки подій елементів керування у формах. Закріплення вмінь і навиків створення форм та елементів керування. Застосування вмінь і навиків програмування алгоритмів лінійної, розгалуженої та циклічної структури.

Завдання.

1. **Створіть форму-калькулятор, подібну до зразка, наведеного нижче, з власним оригінальним дизайном та забезпечте її функціональність. Обов'язково відобразіть у формі вміст комірки пам'яті. У нижній частині форми виведіть інформацію про розробника.** Для цього:



1.1. Створіть новий додаток-форму. Дайте формі ім'я **FormCalc** (властивість **(Neme)**), введіть заголовок, пов'язаний з вашим прізвищем (властивість **Text**) на забезпечте для неї вивід по центру екрана (властивість **StartPosition**);

1.2. Натягніть у формі поля, надпис та кнопки, керуючись зразком, але на власний розсуд;

1.3. Дайте полю для відображення введення і результатів ім'я **textTablo**, а полю вмісту пам'яті – **textMemory**. Забороніть до них доступ (властивість **Enabled**) та задайте вирівнювання по правому краю (властивість **TextAlign**);

1.4. При роботі з пам'яттю в калькуляторі має виконувати попередня операція, а після відображення результатів наступне число має вводитися спочатку, тому створемо **у формі** дві локальні змінні, які будуть відповідати за ці стани:

```
public partial class FormCalc : Form
{
    bool resultTablo = true;
    string operation="";
```

1.5. Оскільки кнопки з цифрами мають додавати відповідні цифри на табло, то для зменшення розміру коду та кількості помилок створимо загальну процедуру додавання цифри на табло:

```
void plusTablo(char symbol)
{   if (resultTablo)
       textTablo.Text = "";
    if (textTablo.Text == "0")
        textTablo.Text = "";
    textTablo.Text += symbol.ToString();
    resultTablo = false;
}
```

,

тоді для введення, наприклад, цифри 9 в процедурі обробки натиснення відповідної кнопки достатньо вказати

```
private void button4_Click(object sender, EventArgs e)
 {plusTablo('9');
 }
```

Самостійно забезпечте функціональність решти кнопок з цифрами.

1.6. Кома між цілою і дробовою частиною може вводитися лише тоді, коли її ще немає. Тому процедура обробки натиснення цієї кнопки може бути така:

```
private void buttonKoma_Click(object sender, EventArgs e)
{   if (resultTablo)
        textTablo.Text = "0";
    bool available = false;
    int i, len;
    len = textTablo.Text.Length;
    for (i = 0; i < len; i++)
        if (textTablo.Text[i] == ',')
        {
            available = true;
            break;
        }
    if (!available)
        textTablo.Text += ",";
    resultTablo = false;
}
```

1.7. **BackSpace** має витирати останній символ, а це можна реалізувати вирізкою підрядка з початку табло:

```
private void button5_Click(object sender, EventArgs e)
{   if (resultTablo)
        textTablo.Text = "0";
    else
        textTablo.Text = textTablo.Text.Substring(0, textTablo.Text.Length - 1);
    if (textTablo.Text == "")
        textTablo.Text = "0";
    resultTablo = false;
}
```

1.8. Створимо тепер процедуру для виконання обчислень безпосередньо над табло, при цьому саму дію передамо параметром підпрограми:

```
private void runOperationTablo(string opr)
{   double tablo;
    if (opr == "")
        return;
    try
    {tablo = Convert.ToDouble(textTablo.Text);
    }
    catch (System.FormatException)
    {   MessageBox.Show("Операцію виконати неможливо", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Error
        textTablo.Text = "0";
        return;
    }
    switch (opr)
    {   case "Sqrt":
            if (tablo < 0)
            {MessageBox.Show("Операція неможлива: корінь з від'ємного числа", "Увага", MessageBoxButtons.O
             return;
            }
            tablo = Math.Sqrt(tablo);
            break;
        case "%":
            tablo *= 0.01;
            break;
```

```
        case "1/x":
            if (tablo == 0)
            {MessageBox.Show("Операція неможлива: ділення на нуль", "Увага", MessageBoxButtons.OK, MessageI
             return;
            }
            tablo = 1 / tablo;
            break;
        case "+/-":
            tablo *= -1;
            break;
    }
    textTablo.Text = tablo.ToString();
    resultTablo = true;
}
```

Тоді для виконання обчисль, пов'язаних лише з табло, достатньо викликати цю процедуру з назвою відповідної дії. Наприклад, процедура обробки натиснення кореня може бутии такою:

```
private void button7_Click(object sender, EventArgs e)
 {runOperationTablo("Sqrt");
 }
```

1.9. І, нарешті, створимо процедуру для виконання обчислень над табло та пам'яттю. При цьому сама дія не передається в процедуру, а виконується попередня збережена операція:

```
private void runOperationMemory()
{   double tablo, memory;
    if (operation == "")
        return;
    try
    {   tablo = Convert.ToDouble(textTablo.Text);
        memory = Convert.ToDouble(textMemory.Text);
    }
    catch (System.FormatException)
    {   MessageBox.Show("Операцію виконати неможливо", "Увага", MessageBoxButtons.OK, MessageBoxIcon.Error)
        textTablo.Text = textMemory.Text = "0";
        return;
    }
    switch (operation)
    {       case "+":
            tablo += memory;
            memory = tablo;
            break;
        case "-":
            tablo = memory - tablo;
            memory = tablo;
            break;
        case "*":
            tablo *= memory;
            memory = tablo;
            break;
        case "/":
            if (tablo == 0)
            {   MessageBox.Show("Операція неможлива: ділення на нуль", "Увага", MessageBoxButtons.OK, Messa
                return;
            }
            tablo = memory / tablo;
            memory = tablo;
            break;
        case "M+":
            memory += tablo;
            break;
        case "M-":
            memory -= tablo;
            break;
        case "MC":
            memory = 0;
            break;
        case "MR":
            tablo = memory;
            break;
        case "MT":
            memory = tablo;
            break;
        case "C":
            tablo = 0;
            memory = 0;
            break;
    }
```

```
    operation = "";
    textTablo.Text = tablo.ToString();
    textMemory.Text = memory.ToString();
    resultTablo = true;
}
```

Тоді при натисненні, наприклад, кнопки додавання потрібно спочатку **виконати попередню операцію**, потім **результат зчитати з пам'яті в табло**, а + запам'ятати для наступного виконання:

```
private void buttonPlus_Click(object sender, EventArgs e)
{   runOperationMemory();
    operation = "MT";
    runOperationMemory();
    operation = "+";
}
```

1.10. Функціональність решти кнопок забезпечте замостійно

2. **Доповніть форму-калькулятор двома функціональними кнопками, відмінними від кнопок одногрупників, для виконання двох корисних функцій (наприклад, піднесення до степеня) та забезпечте їх дієздатність.**
3. **Опублікуйте вашу форму-калькулятор в загальному репозиторії потоку на сайті github.com. Для цього:**
   3.1. Перейменуйте вашу форму-калькулятор і пов'язані з нею файли до формату **FormCalcXX**, де XX – ваше прізвище англійською мовою;
   3.2. Клонуйте на власний ПК віддалений репозитарій https://github.com/IPZMEGU/Calc-20XX (де XX - останні дві цифри поточного року) з облікового запису IPZMEGU (пароль: ipzmegu1999). Якщо пристрій, з якого здійснюється вхід, новий для цього облікового запису, то для реєстрації такого пристрою введіть код підтвердження, який міститься у листі на пошті IPZMEGU@gmail.com (пароль входу в пошту: ipzmegu1999);
   3.3. Додайте в клонований проект вашу форму-калькулятор ;
   3.4. В клонованому проекті у формі **FormStart** створіть кнопку з надписом з шифру вашої групи та прізвища та забезпечте при її натисненні виклик доданої форми-калькулятора;
   3.5. Синхронізуйте клонований проект з віддаленим репозитарієм.