



Cel i zakres zajęć:

Głównym celem zajęć jest zapoznanie się z właściwościami i zdarzeniami podstawowych komponentów w środowisku *Microsoft Visual C++ 2010 Express Edition*. Przeprowadzenie ćwiczenia pozwala na poznanie projektów typu *Windows Forms Application*, naukę składni podstawowych komponentów w interakcji z użytkownikiem.

Przebieg ćwiczenia:

1. Proszę stworzyć nowy projekt typu: *Windows Forms Application*.
2. Zbudować szkielet graficzny formatki zgodnie z rysunkiem nr 1, w którym wykorzystano następujące komponenty: *Button*, *TextBox*, *Label*, *LinkLabel* oraz okna dialogowe: *MenuStrip1*, *OpenFileDialog1*, *ContextMenuStrip1*, *ColorDialog1*, *FontDialog1*.

Form1

Plik Zamknij

INPUT SCORE Memory 0

7 8 9 / MR

4 5 6 * MC

1 2 3 - M+

0 +/- . + =

ToDo Change color Change font www.wp.pl

Rys.1.Graficzna reprezentacja głównej formatki programu.

3. Zadeklarować globalne zmienne (operator :: oznacza przynależność do klasy):

```
Double liczba, liczba_1,liczba_2;  
Double suma, roznica, iloczyn, iloraz;  
Double bufor;  
Char operacja;  
Boolean przecinek;
```

4. Ustawić wartość zmiennej bufor na zero w zdarzeniu wczytywania formatki (operator ^ jest odpowiednikiem referencji):

```
private: System::Void Form1_Load(System::Object^ sender,  
    System::EventArgs^ e)  
{  
    bufor=0;  
}
```

5. Pod odpowiednimi przyciskami klawiatury kalkulatora dodać właściwą im funkcjonalność (dopisanie cyfry do *TekstBoxa* oznaczonego *label*em „INPUT”):

```
Np.   textBox4->Text=textBox4->Text+"0";
```

6. Oprogramować przycisk separatora liczby zmiennoprzecinkowej w taki sposób, aby uniemożliwić użytkownikowi wprowadzenie przecinka jako pierwszego znaku oraz wykluczyć kilkukrotne użycie separatora.

```
if (przecinek==false)  
{  
    if (textBox4->Text!="")  
    {  
        textBox4->Text=textBox4->Text+",";  
        przecinek=true;  
    }  
}
```

7. Dodać funkcjonalność klawisz zmieniającego znak wprowadzonej liczby na przeciwny.

```
liczba=Convert::ToDouble(textBox4->Text);  
if (liczba>0)  
{  
    textBox4->Text="-"+textBox4->Text;  
}  
if (liczba<0)  
{  
    liczba=liczba*(-1);  
    textBox4->Text=liczba.ToString();  
}
```

8. Wykorzystując zmienna *bufor*, dodać funkcjonalność przyciskom sterującym pamięcią programu. Zadbać o to, aby w czasie przechowywania zmiennej w pamięci wyświetlana była jej zawartość w dedykowanym *labelu* obok słowa *Memory*.

```
MR:   textBox4->Text=bufor.ToString();  
MC:   bufor=0;  
      label5->Text=bufor.ToString();
```

```
M+:   bufor=bufor+Convert::ToDouble(textBox4->Text);
      label5->Text=bufor.ToString();
```

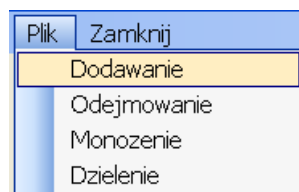
9. Dla przycisków będących reprezentantami poszczególnych operacji matematycznych (+,-,*,/) zaimplementować stosowne instrukcje kodu. Przykład dodawania:

```
if (textBox4->Text!="")
{
    liczba_1=Convert::ToDouble(textBox4->Text);
    operacja='+';
    textBox4->Text="";
    przecinek=false;
}
else
{
    MessageBox::Show("Wprowadz liczbe!!",
        "Błąd",MessageBoxButtons::OK,MessageBoxIcon::Error);
}
```

10. Dla przycisku "=" dodać kod zapewniający wykonanie wybranej operacji przy zachowaniu elementarnych zabezpieczeń pustego pola i dzielenia przez zero.

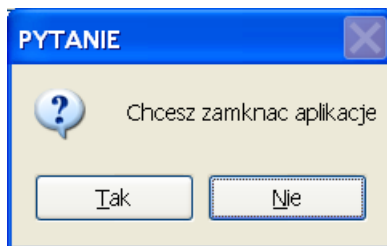
```
if (operacja=='/')
{
    if (Convert::ToDouble(textBox4->Text)!=0)
    {
        liczba_2=Convert::ToDouble(textBox4->Text);
        iloraz=liczba_1/liczba_2;
        textBox5->Text=iloraz.ToString();
        przecinek=false;
        textBox4->Text="";
    }
    else
    {
        MessageBox::Show("Nie dziel przez zero !!",
            "Błąd",MessageBoxButtons::OK,MessageBoxIcon::Error);
    }
}
```

11. Zbudować menu główne aplikacji pozwalające na wykonanie operacji matematycznych zaimplementowanych wcześniej pod właściwymi przyciskami (Rys.2). Funkcja zamknij pozwala na opuszczenie aplikacji po uprzednim zatwierdzeniu właściwego komunikatu.



Rys.2. Struktura menu programu.

W momencie wywołania akcji „Zamknij” użytkownikowi pojawia się okno typu *MessageBox* przedstawione na rysunku nr 3, w którym widnieje ikona ostrzeżenia i domyślnie zaznaczony jest przycisk „Nie”.



Rys.3. Widok okna opuszczenia programu.

```
System::Windows::Forms::DialogResult odp =
MessageBox::Show("Chcesz zamknąć aplikację", "PYTANIE",
MessageBoxButtons::YesNo, MessageBoxIcon::Question,
MessageBoxDefaultButton::Button2);
switch (odp)
{
case System::Windows::Forms::DialogResult::Yes:
    Close();
case System::Windows::Forms::DialogResult::No:
    break;
}
```

12. W celu załadowania pliku tekstowego `ToDo.txt` do okna dedykowanego `textBoxa`, należy uprzednio wykonać dwa kroki:
 - zmienić właściwość *MultiLine* komponentu *TextBox* na `TRUE`
 - dołączyć przestrzeń nazw wejścia/wyjścia w programie

```
using namespace System::IO;
```

Przykład instrukcji spod klawisza *ToDo*:

```
openFileDialog1->Filter="Pliki tekstowe (*.txt)|*.txt|Wszystkie  
pliki (*.*)|*.*)";

openFileDialog1->ShowDialog();
StreamReader^ plik = gcnew StreamReader(
openFileDialog1->FileName, System::Text::Encoding::Default);

textBox1->Text=plik->ReadToEnd();
plik->Close();
```

13. Pod klawiszem *ChangeColor* należy uzupełnić kod w taki sposób, aby odwołując się do komponentu *ColorDialog* umożliwić zmianę tła w *TextBoxie* wyświetlającym zawartość pliku *ToDo.txt*.

```
if (colorDialog1->ShowDialog() ==
    System::Windows::Forms::DialogResult::OK)
{
    textBox1->BackColor=colorDialog1->Color;
}
```

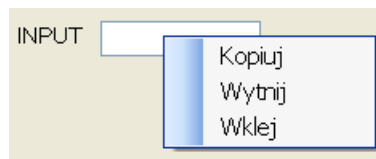
14. Pod klawiszem *ChangeFont* należy uzupełnić kod w taki sposób, aby odwołując się do komponentu *FontDialog* umożliwić zmianę czcionki w *TextBoxie* wyświetlającym zawartość dowolnego pliku tekstowego.

```

if(fontDialog1->ShowDialog()==
    System::Windows::Forms::DialogResult::OK)
{
    textBox1->Font=fontDialog1->Font;
}

```

15. Używając komponentu *ContextMenuStrip*, zdefiniować menu podręczne dla komponentu *TextBox* oznaczonego *label*em "INPUT". Menu podręczne należy skojarzyć z *TextBox*em za pomocą stosownej właściwości tego komponentu (*ContextMenuStrip*).



Rys.4. Menu podręczne komponentu *TextBox*.

Funkcjonalność menu określają trzy funkcje: Kopiuż, Wytnij oraz Wklej, obsługujące schowek aplikacji.

```

textBox4->Copy();
textBox4->Cut();
textBox4->Paste();

```

16. W dolnej części formatki umieszczono komponent *LinkLabel* umożliwiający bezpośrednie wywołanie okna przeglądarki z poziomu aplikacji. Dla komponentu należy określić właściwość *Text* jako www.wp.pl

```

private: System::Void linkLabel1_LinkClicked(System::Object^
    sender, System::Windows::Forms::LinkLabelLinkClickedEventArgs^ e)
{
    System::Diagnostics::Process::Start(linkLabel1->Text);
    linkLabel1->LinkVisited=true;
}

```