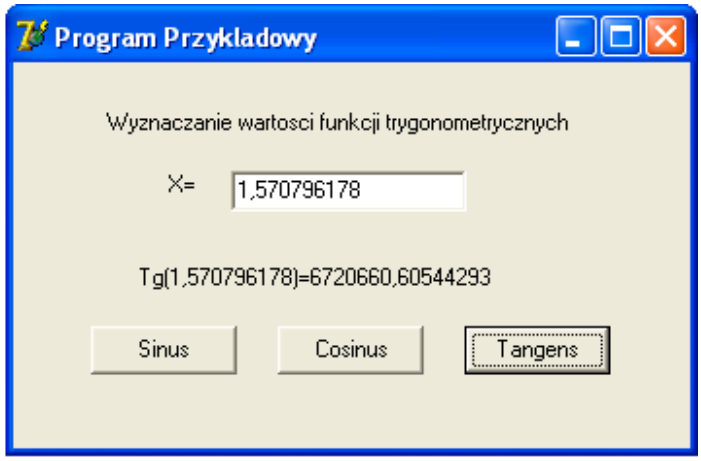
Wydział: ZiMK
Przedmiot: Programowanie - Delphi Semestr 5

Ćwiczenie 5 - Komponent Edit. Okienka komunikatów.

1. Wyznaczanie wartości funkcji trygonometrycznych

Zrealizuj projekt aplikacji służącej do wyznaczania wartości funkcji sin, cos i tg.
Postać wyświetlanego okienka (w fazie wykonania programu):



- Rozmieść ma formatce trzy etykiety, pole edycyjne **Edit** oraz 3 przyciski **Button**. Ustaw zgodnie z rysunkiem właściwości **Caption** formatki, dwu etykiet i przycisków.

- Zdefiniuj obsługę zdarzenia **OnClick** dla przycisku:

```
procedure TForm1.Button1Click(Sender: TObject);
var f:real;
begin
f:=sin(StrToFloat(Edit1.Text));
Label3.Caption:='Sin('+Edit1.Text+')='+FloatToStr(f);
end;
```

  W zapisie metody do konwersji tekstów wpisywanych w polu edycyjnym **Edit** (właściwość **Text**) zastosowano funkcje konwersji typu:
  StrToFloat - konwersja łańcucha na liczbę rzeczywistą,
  FloatToStr - konwersja liczby rzeczywistej na łańcuch

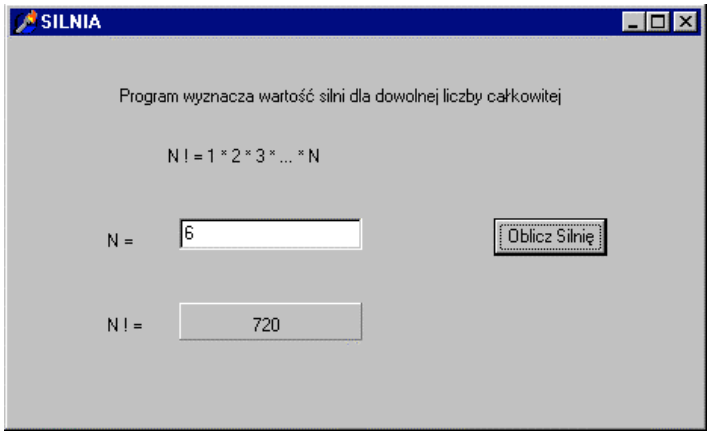
- Przetestuj działanie przycisku *Sinus*, a następnie samodzielnie zdefiniuj działanie przycisków *Cosinus* i *Tangens*.
  Uwaga. Obliczanie funkcji tangensa trygonometrycznego z zastosowaniem funkcji **tan** wymaga dołączenia do aplikacji modułu **Math**, należy więc odpowiednio zmodyfikować dyrektywę **uses** modułu Unit1 - dopisać nazwę Math.

- Zmodyfikuj aplikację w ten sposób, żeby wartości argumentu funkcji można było określać w stopniach, a nie w radianach.

- Sprawdź, czy aplikacja działa poprawnie gdy zamiast liczby użytkownik wpisze w polu edycyjnym dowolny napis. Jak poprawić aplikację?

2. Zrealizuj projekt aplikacji służącej do wyznaczania wartości silni.

Postać wyświetlanego okienka:



- Rozmieść w oknie aplikacji: komponent **Panel**, 5 etykiet, przycisk **Button** oraz komponent edycyjny **Edit**. Ustal odpowiednio początkowe właściwości komponentów.

- Zdefiniuj obsługę zdarzenia **OnClick** dla przycisku:

```
procedure TForm1.Button1Click(Sender: TObject);
var i, N, Silnia: Integer;
begin
  try
     N:=StrToInt(Edit1.Text);

     {tu obliczenie wartości Silnia = N! }

     Label5.Caption:=IntToStr(Silnia);
  except
       MessageDlg('Uwaga! Błąd', mtError, [mbOK], 0);
       Edit1.SetFocus;
  end;
end;
```

UWAGA: Przydatne procedury do konwersji tekstów wpisywanych w polu edycyjnym *Edit* (własność *Text*):

StrToInt - konwersja łańcucha na liczbę całkowitą,
IntToStr - konwersja liczby całkowitej na łańcuch

Przydatne procedury wyświetlania komunikatów (poniżej nagłówki procedur):

function MessageDlg(**const** Msg: String; AType: TMsgDlgType;
AButtons: TMsgDlgButtons; HelpCtx: LongInt): Word;

Msg - napis w okienku
AType - typ okienka może przyjmować wartości: - *mtWarning* (!), *mtError* (STOP), *mtInformation* (i), *mtConfirmation* (?), *mtCustom* - bez bitmapy (tytuł - nazwa pliku programu),
AButton - zbiorowy - określa jakie przyciski będą występować w okienku. Dopuszczalne wartości: *mbYes*, *mbNo*, *mbOK*, *mbCancel*, *mbHelp*, *mbAbort*, *mbRetry*, *mbIgnore*, *mbAll*.
HelpCtx - określa który ekran Helpu jest dostępny w czasie wyświetlania okienka.

Funkcja zwraca wartość przyciśniętego klawisza: *mrNone*, *mrOK*, *mrCancel* itp.

function MessageDlgPos(**const** Msg: String; AType: TMsgDlgType; AButtons: TMsgDlgButtons; HelpCtx: LongInt; X,Y: Integer): Word;

jw. dodatkowo *X* i *Y* wyznacza miejsce okienka (poprzednie wyświetlane jest centralnie).

- Przetestuj działanie aplikacji.

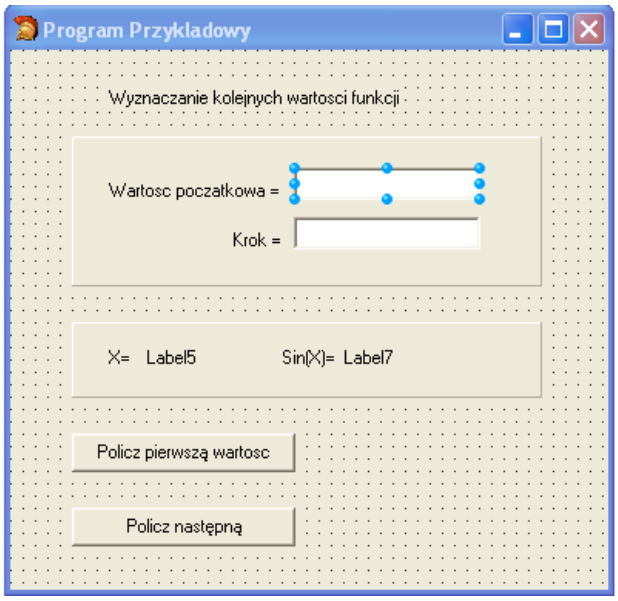
3. Popraw aplikację zapisaną w zadaniu pierwszym (wyznaczanie funkcji trygonometrycznych) w ten sposób, żeby wpisanie niepoprawnej wartości argumentu nie powodowało błędu wykonania lecz jedynie wyświetlenie odpowiedniego komunikatu.

- Dodaj metodę obsługi zdarzenia OnExit dla komponentu Edit1 postaci:

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
  try
    StrToFloat(Edit1.Text);
  except
    MessageDlg('Uwaga! Błąd', mtError, [mbOK], 0);
    Edit1.SetFocus;
  end;
end;
```

4. Zrealizuj aplikację wyznaczającą kolejne wartości funkcji sinus.

Postać wyświetlanego okienka (w fazie projektowania aplikacji):



- Rozmieść w oknie aplikacji: etykietę tytułową, dwa przyciski, 2 panele. Na pierwszym panelu rozmieść dwie etykiety i dwa pola edycyjne **Edit**, a na drugim 4 etykiety. komponent **Panel**, 5 etykiet, przycisk **Button** oraz komponent edycyjny **Edit**. Ustal początkowe właściwości komponentów zgodnie z rysunkiem. Wyłącz aktywność przycisku drugiego - właściwość **Enabled** tego przycisku ustaw na *False*.

- Zdefiniuj obsługę zdarzeń **OnExit** dla obu pól edycyjnych. Proponowany zapis dla pola wartości początkowej:

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
try
  StrToFloat(Edit1.Text);
except
  MessageDlg('Uwaga! Błąd', mtError, [mbOK], 0);
  Edit1.SetFocus;
end;
end;
```

  dla pola kroku:

```
procedure TForm1.Edit2Exit(Sender: TObject);
begin
try
  StrToFloat(Edit2.Text);
  Button2.Enabled:=True;
except
  MessageDlg('Uwaga! Błąd', mtError, [mbOK], 0);
  Edit2.SetFocus;
end;
end;
```

  Przetestuj działanie wprowadzonych metod.

- W sekcji **public** definicji typu formatkowego *TForm1* dodaj nowe pole:

```
   x: real
```

  przeznaczone do przechowywania bieżącej wartości argumentu funkcji.

- Dodaj obsługę zdarzenia **OnClick** dla przycisku "policz pierwszą wartość":

```
procedure TForm1.Button1Click(Sender: TObject);
begin
x:=StrToFloat(Edit1.Text);
Label5.Caption:=Edit1.Text;
Label7.Caption:=FloatToStr(sin(x));
end;
```

  Przetestuj działanie przycisku.

- Opracuj samodzielnie i dołącz metodę obsługi zdarzenia OnClick dla przycisku "policz następną". Przetestuj działanie metody i całej aplikacji.