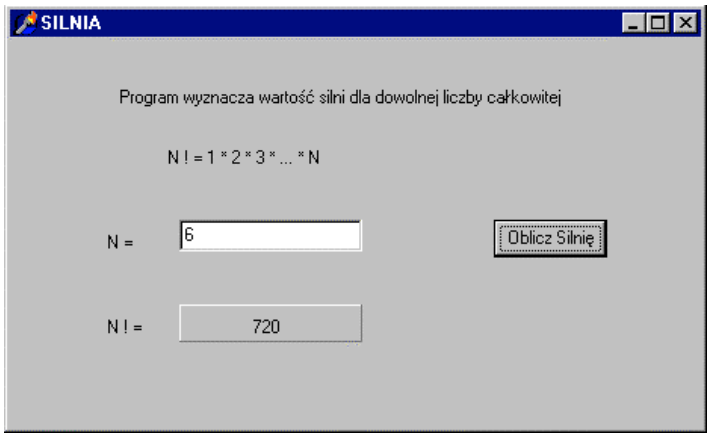
Wydział: **Zarządzania i Modelowania Komputerowego**
Przedmiot: **Języki programowania - Delphi**
Rok **III** Semestr 5
Opracował: **dr inż. Krzysztof Strzałkowski**

Ćwiczenie 6
Komponent Edit. Okienka komunikatów.

1. Wyznaczanie wartości silni

Zrealizuj projekt aplikacji służącej do wyznaczania wartości silni.
Postać wyświetlanego okienka:



- Rozmieść w oknie aplikacji: komponent **Panel** (dla wyświetlania na nim wyniku obliczeń), 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, kod: Integer;
begin
Val(Edit1.Text, N, kod);
if kod = 0 then
   begin

   {tu obliczenie wartości Silnia = N! }

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

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

StrToFloat	- konwersja łańcucha na liczbę rzeczywistą,
FloatToStr	- konwersja liczby rzeczywistej na łańcuch
StrToInt	- konwersja łańcucha na liczbę całkowitą,
IntToStr	- konwersja liczby całkowitej na łańcuch
Val(łańcuch, x, kod)	- zamiana *łańcucha* na liczbę *x*. Jeśli postać zapisu liczby w łańcuchu jest błędna, to *kod* (<>0) podaje numer znaku błędnego.

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.

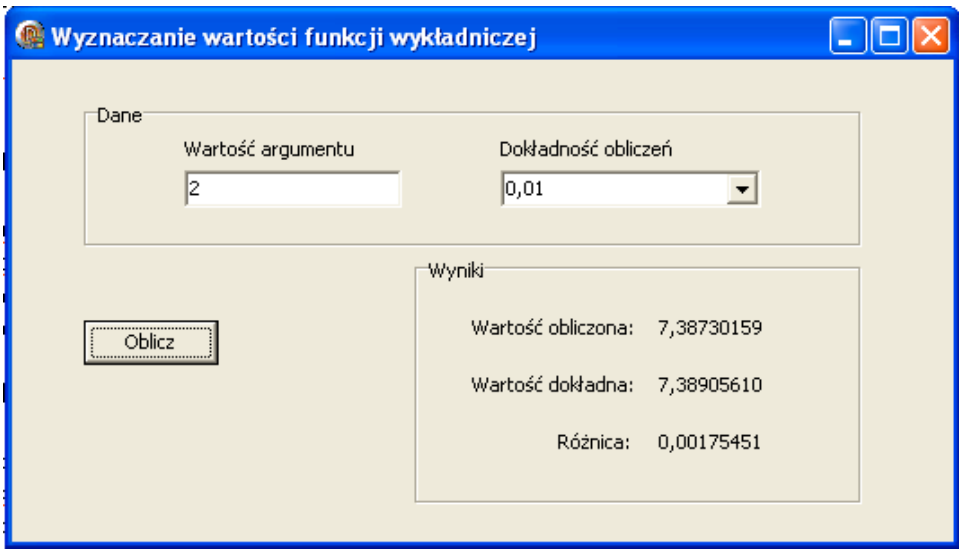
function MessageDlgPos(**const** Msg: String; AType: TMsgDlgType; AButtons:
TMsgDlgButtons; HelpCtx: LongInt; X,Y: Integer): Word;

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

- Zapisz projekt w oddzielnym katalogu i przetestuj działanie aplikacji.

2. Wyznaczanie wartości funkcji wykładniczej

Zrealizuj projekt aplikacji służącej do wyznaczania wartości funkcji e^x.
Postać wyświetlanego okienka:



Po wpisaniu wartości argumentu **x** i wybraniu żądanej dokładności obliczeń **eps** oraz po naciśnięciu przycisku **Oblicz** aplikacja powinna wyznaczać przybliżoną wartość funkcji e^x na podstawie szeregu:

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

Należy sumować tyle kolejnych wyrazów szeregu by pierwszy pominięty wyraz był mniejszy od dokładności **eps**.

- Rozmieść w oknie aplikacji przycisk **Button** i dwa komponenty **GroupBox**. Na komponencie GroupBox zatytułowanym **Dane** rozmieść komponent **Edit** i komponent **ComboBox**. Na drugim komponencie GroupBox rozmieść 6 etykiet - trzy z nich przeznaczone na wyniki obliczeń. Ustaw wg rysunku właściwości **Caption** komponentów i formatki. Dla komponentu **Edit** ustaw właściwość **Text** równą **0**.
  W komponencie ComboBox ustaw listę dostępnych wartości dokładności obliczeń. W tym celu wykorzystując edytor właściwości **Items** (przycisk - trzy kropki) wpisz listę:

  0,1
  0,05
  0,01
  0,005
  0,001
  0,0001

  Ustaw także właściwość **ItemIndex** równą **0**.
  Dla właściwości **Style** wybierz wartość **csDropDownList**

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

```
procedure TForm1.Button1Click(Sender: TObject);
var x, eps, s, w: real;
      i: integer;
begin
{ Ustalenie wartości zmiennych x oraz eps na podstawie zapisów we
właściwości Text komponentu Edit oraz właściwości Text komponentu ComboBox}
{ Zbadanie poprawności wprowadzenia liczby x. W przypadku błędnego wpisu
wyświetlenie odpowiedniego komunikatu i zakończenie metody. W przypadku
wprowadzenia poprawnej wartości wykonanie obliczeń jak poniżej }

s:=1;
w:=1;
i:=1;
repeat
  w:=x*w/i;
  i:=i+1;
  s:=s+w;
until abs(w)<eps;

{ wyprowadzenie wyników obliczeń: wartości obliczonej, dokładnej wartości
funkcji exp(x) oraz różnicy tych dwu wyników }
end;
```

  Wyniki obliczeń należy wyprowadzać w sposób sformatowany z wykorzystaniem funkcji:

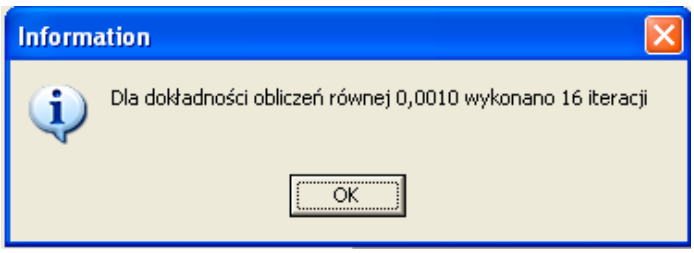
  function FloatToStrF(Value: Extended; Format: TFloatFormat;
  Precision, Digits: Integer): string;

Value	liczba którą należy przedstawić w postaci sformatowanego napisu
Format	typ formatowania – można wpisywać wartości: ffGeneral, ffExponent, ffFixed, ffNumber, ffCurrency. Przy wpisaniu **ffFixed** pozostałe dwa parametry mają znaczenie:
Precision	ilość wszystkich cyfr znaczących w wynikowym przedstawieniu liczby
Digits	ilość cyfr części ułamkowej

  Wynikiem funkcji jest napis wyrażający liczbę w postaci sformatowanej.

- Zapisz projekt w oddzielnym katalogu i przetestuj działanie aplikacji.

- Uzupełnij projekt o kolejny przycisk powodujący wyświetlanie okienka komunikatu zawierającego liczbę cykli pętli wykonanych przy ostatnim obliczeniu funkcji.
  Przykładowa postać okienka:



3. Zadanie do samodzielnego opracowania

Opracuj aplikację do rozwiązywania metodą iteracji prostej równania:

x – cos(x) = 0

Dla znanych wartości przybliżenia początkowego **x0** (dowolna liczba) oraz dokładności obliczeń **eps** kolejne przybliżenia rozwiązania należy wyznaczać w pętli:

```
repeat
  x := x0;
  x0 := cos( x0 );
until abs( x - x0 ) < eps;
```

Opracowana aplikacja powinna wykorzystywać komponenty:

Edit	- do wprowadzania wartości przybliżenia początkowego *x0*,
ComboBox	- dla wyboru dokładności obliczeń *eps*,
Buton	- do inicjowania obliczeń (po sprawdzeniu poprawności liczby *x0*)

Uzupełnij aplikację o nowy przycisk wyświetlający w okienko komunikatu liczbę wykonanych cykli pętli.