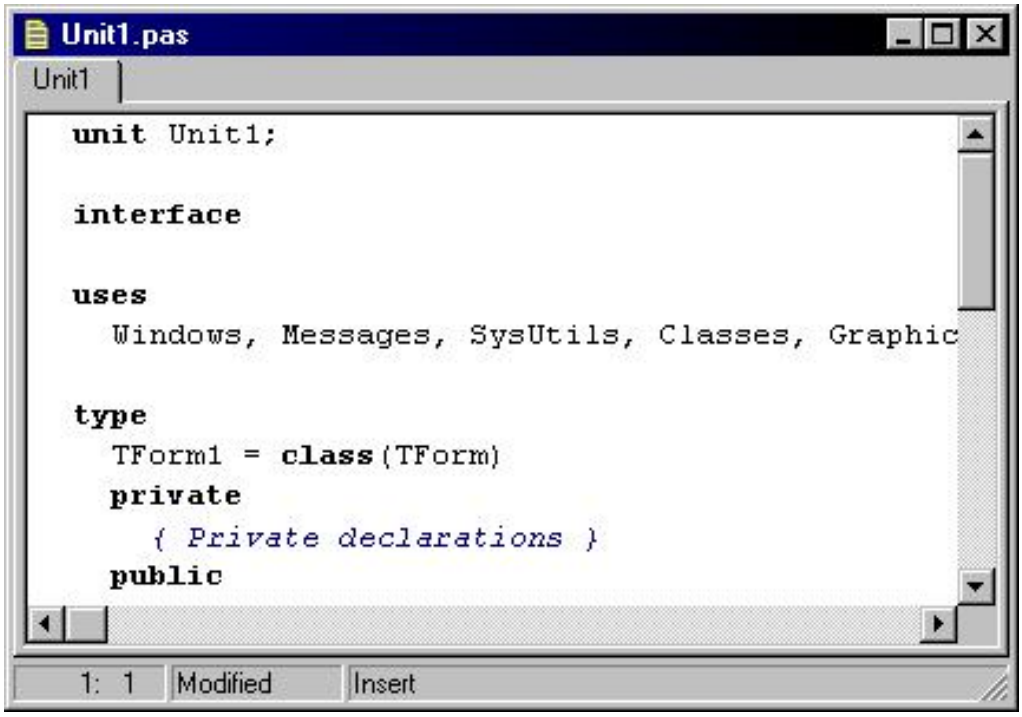
Wstęp	Czym jest Delphi. W jaki sposób wstawiać kody i komponenty. Jak korzystać z poniższego kursu.
Lekcja 1	Ćwiczenie z wykorzystaniem zmiennych i komponentu TGauge
Lekcja 2	Operacje na liczbach (dodawanie, odejmowanie, dzielenie, mnożenie)
Lekcja 3	TTimer co pewien czas. Gra
Lekcja 4	A jednak odlicza. Stoper
Lekcja 5	Aktualny czas, czyli zegar
Lekcja 6	Tworzymy budzik
Lekcja 7	Wyświetlamy nowe okno
Lekcja 8	Czas na grafikę
Lekcja 9	Rysujemy koło o wybranym kolorze
Lekcja 10	Nazwa wybranego koloru w TEdit
Lekcja 11	Prosta animacja
Lekcja 12	Podmiana obrazka po najechaniu muszą [OnMouseMove]
Lekcja 13	Zdarzenie wygenerowane po wciśnięciu wybranego klawisza [OnKeyDown]
Lekcja 14	try...except czyli obsługa wyjątków
Lekcja 15	*For I := 0 to* ... (pętle)
Lekcja 16	Przeglądarka graficzna
Lekcja 17	Zabawa na palecie kolorów

Czym jest delphi ?

Delphi jest środowiskiem programowaania RAD(Rapid Applicaation Development). Służy do szybkiego tworzenia 32-bitowych aplikacji działających pod Windows. Tworząc swoje programy w tym Delphi będziesz wykorzystywał język Object Pascal. Całe programowanie staje się prostrze i szybsze dzięki bibliotece VCL.

Jak wpisywać kody ?



Kody wpisujemy w oknie podobnym do tego obok. Zazwyczj muszą zostać wpisane wewnątrz obsługi danego zdarzenia, np. przy kliknięciu na przycisk jest generowane zdarzenie OnClick. Jednak czasami nie wystarczy wpisać kod. Załóżmy, że w kursie otrzymasz polecenie podwójnego kliknięcia na przycisk i dodania kodu:

```
var
X : Integer;
Begin
X := X + X;
```

Wówczas kod całego zdarzenia będzie przypomminał poniższy:

```
procedure TForm1.Button1Click(Sender: TObject);
Begin
var
X : Integer;
Begin
X := X + X;
end;
```

Jak pewnie zauważyłeś przed znacznikiem **var** znajduję sie napis **Begin**. Jest to niedopuszcalne i przy próbie uruchomienia programu zostanie wyświetlony błąd. Należy wtedy usunąć słowo Begin znajdujące się przed słowem Var.

Jak wstawiać komponenty ?

Wstawianie komponentów przypomina rysowanie prostokątów w programie graficznym. Wystarczy poprzez kliknięcie na palecie komponentów wybrać jeden interesujący nas obiekt. Następnie na szablonie programu należy narysować jego kontur.

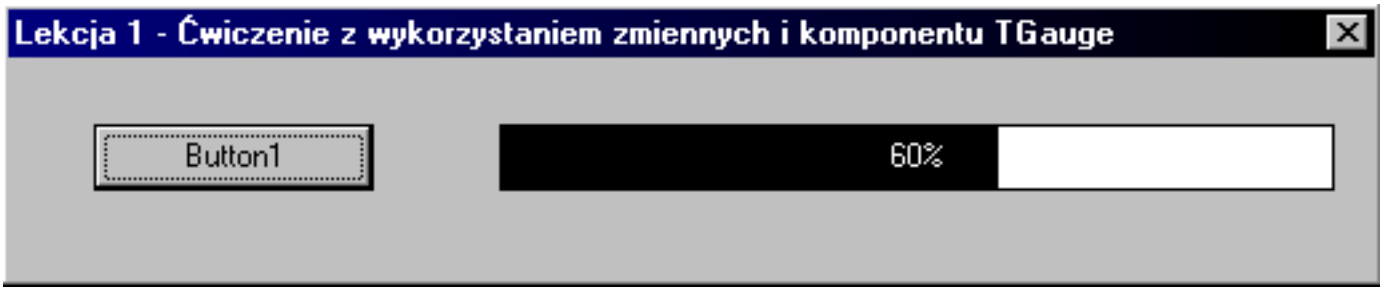
Jak korzystać z kursu ?

Na początku każdego kursu znajduje się opis tworzonego programu. Zaraz pod nim znajdziesz obrazek ilustrujący zakończony program. Możesz nim się sugerować przy wsawianiu komponentów z tabeli. Po zrobieniu tej czynności należy wykonać etapy znajdujące się w sekcji **Medody**. Jeżeli jakiś etap wstawiania kodu był nie jasny, możesz go sprawdzić w sekcji **Kod źródłowy**

Cel :

Chcemy uzyskać efekt, aby po kliknięciu na przycisk powiększyć o " 10 " wartość Progress komponentu TGauge.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Gauge	TGauge
Button1	TButton

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Do obsługi funkcji OnClick przycisku dodajemy poniższy kod:

```
procedure TForm1.Button1Click(Sender: TObject);
var
X : Integer;
begin
```

```
X := Gauge.Progress;
X := X + 10;
Gauge.Progress := X ;
end;
```

3) Uruchamiamy program.

Kod źródłowy :

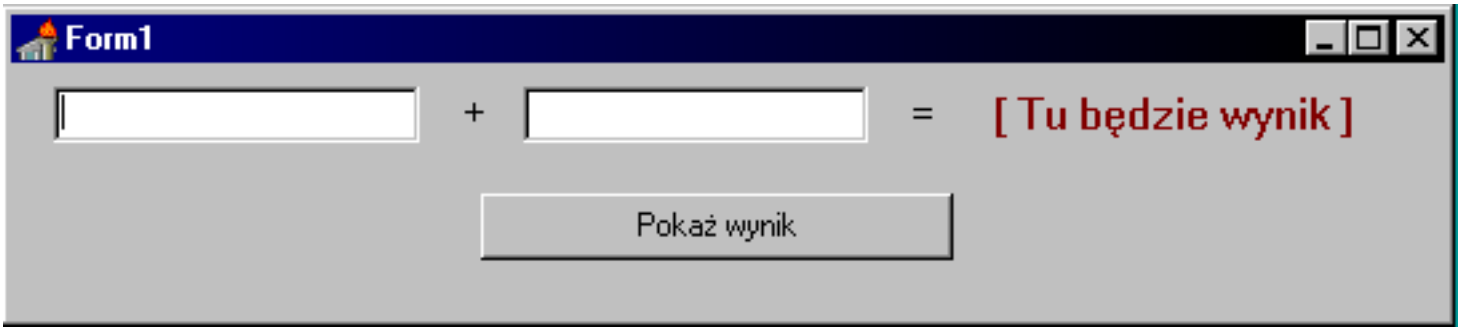
```
unit Unit1;

interface
uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
Gauges, StdCtrls;

type
TForm1 = class(TForm)
Button1: TButton;
Gauge1: TGauge;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
X : Integer;
begin
```

```
X := Gauge.Progress;
X := X + 10;
Gauge.Progress := X ;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po wpisaniu liczb w oba pola i wciśnięciu przycisku została wyświetlona suma tych liczb.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Edit1	TEdit
Edit2	TEdit
Label1	TLabel
Label2	TLabel
Label3	TLabel
Button1	TButton

Metoda :

1) Wstawiamy komponenty, wypisane.w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Zmieniamy właściwości Caption lub Text według własnego uznania lub sugerując się rysunkiem

3) Do obsługi funkcji OnClick przycisku dodajemy poniższy kod:

```
procedure TForm1.Button1Click(Sender: TObject);
var
X, Y, Z : Integer;
begin
X := StrToInt(Edit1.text);
Y := StrToInt(Edit2.text);
Z := X + Y;
Label3.Caption := IntToStr(Z);
```

4) Uruchamiamy program.

Porady :

Zmieniając znak działania na +, -, /, * zmienia się sposub wykonywania działania.

Kod źródłowy :

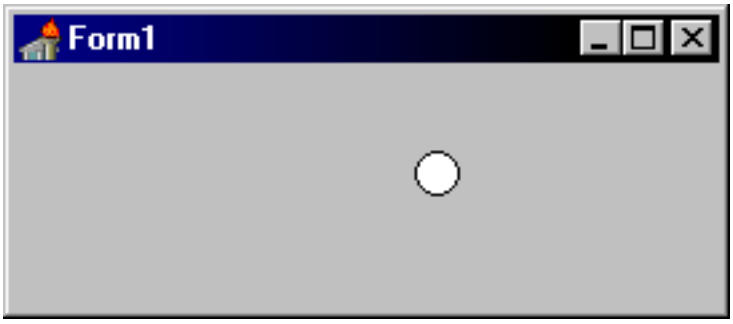
```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Edit1: TEdit;
Label1: TLabel;
Edit2: TEdit;
Label2: TLabel;
```

```
Label3: TLabel;
Button1: TButton;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
X, Y, Z : Integer;
begin
X := StrToInt(Edit1.text);
Y := StrToInt(Edit2.text);
Z := X + Y;
Label3.Caption := IntToStr(Z);
end;

end.
```

Cel :

Chcemy uzyskać efekt przesuwającej się piłeczki po pulpicie naszego programu, wzdłuż drogi poziomej.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Timer1	TTimer
Kolo	TShape

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Klikamy na puste pole formularza i w zakładce Events klikamy dwa razy na zdarzenie OnActive. Dopisujemy poniższy kod:

```
prawo := true;
```

3) W kodzie źródłowym znajdujemy napis " implementation ". Pod nim wpisujemy poniższy kod:

```
var
prawo : Boolean;
```

4) Do obsługi zdarzenia OnTimer komponentu Timer1 dodajemy kod:

```
procedure TForm1.Timer1Timer(Sender: TObject);
var
X : integer;
begin
if prawo = true then Kolo.left := kolo.left + 10;
if prawo = false then Kolo.left := kolo.left - 10;
if kolo.left = 0 then prawo := true;
X := form1.width - kolo.width - kolo.width;
if kolo.left >= X then prawo := false;
end;
```

5) Właściwość Interval komponent Timer1 ustawiamy na 100

6) Uruchamiamy program

Kod źródłowy :

```
unit Unit1;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls;

type
TForm1 = class(TForm)
Kolo: TShape;
Timer1: TTimer;
procedure Timer1Timer(Sender: TObject);
procedure FormActivate(Sender: TObject);
```

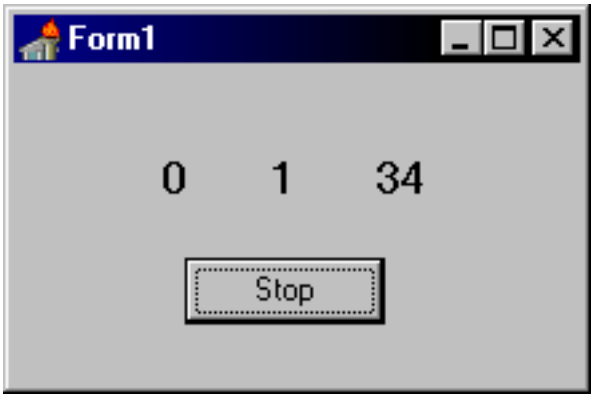
```
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

var
prawo : Boolean;

procedure TForm1.Timer1Timer(Sender: TObject);
var
X : integer;
begin
if prawo = true then Kolo.left := kolo.left + 10;
if prawo = false then Kolo.left := kolo.left - 10;
if kolo.left = 0 then prawo := true;
X := form1.width - kolo.width - kolo.width;
if kolo.left >= X then prawo := false;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
prawo := true;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po kliknięciu na przycisk był odliczany czas, aż do jego zatrzymania. Przypominac to będzie stoper

---

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Button1	TButton
Secundy	TLabel
Minuty	TLabel
Godziny	TLabel
Timer1	TTimer

Metoda :

1) Wstawiamy komponenty, wypisane.w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Ustawiamy właściwości Caption komponentów Label na: 0, zaś komp. TButton na: Stop

3) Do obsługi funkcji OnTimer dodajemy poniższy kod:

```
X := StrToInt(Secundy.Caption) + 1;
if StrToInt(Secundy.Caption) = 59 then Begin
X := 0;
Y := Y + 1;
Minuty.Caption := IntToStr(Y);
end;
if StrToInt(Secundy.Caption) = 59 then Begin
if StrToInt(Minuty.Caption) = 59 then Begin
Y := 0;
Minuty.Caption := IntToStr(Y);
Z := Z + 1;
Godziny.Caption := IntToStr(Z);
end;
end;
Secundy.Caption := IntToStr(X);
```

4) Znajdujemy w kodzie napis **implementation**, i dodajemy pod nim deklaracje globalną zmiennych:

```
var
X,Y, Z : integer;
```

5) Do obsługi funkcji OnClick komp. Button dodajemy kod:

```
if Timer1.Enabled = True then begin
Timer1.Enabled := false;
Button1.Caption := 'Start';
end else begin
Timer1.Enabled := true;
Button1.Caption := 'Stop';
end;
```

6) Klikamy pojedyńczo w puste miejsce formularza i z zakładki Event wybieramy funkcję **OnActive**. Wpisujemy kod:

X := 0;
Y := 0;
Z := 0;

7) Uruchamiamy program.

Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ExtCtrls;

type
TForm1 = class(TForm)
Timer1: TTimer;
Secundy: TLabel;
Minuty: TLabel;
Godziny: TLabel;
Button1: TButton;
procedure Timer1Timer(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
{ Private declarations }
public

{ Public declarations }
end;

var
Form1: TForm1;

implementation
```

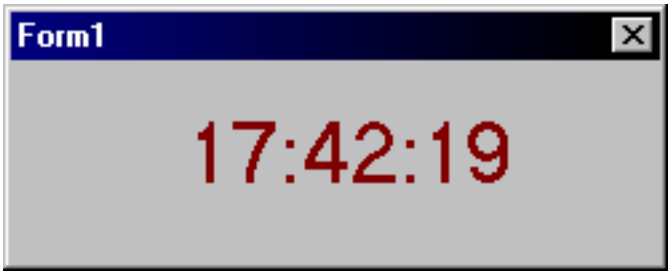
```
{$R *.DFM}

var
X,Y, Z : integer;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
X := StrToInt(Secundy.Caption) + 1;
if StrToInt(Secundy.Caption) = 59 then Begin
X := 0;
Y := Y + 1;
Minuty.Caption := IntToStr(Y);
end;
if StrToInt(Secundy.Caption) = 59 then Begin
if StrToInt(Minuty.Caption) = 59 then Begin
Y := 0;
Minuty.Caption := IntToStr(Y);
Z := Z + 1;
Godziny.Caption := IntToStr(Z);
end;
end;
Secundy.Caption := IntToStr(X);
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
if Timer1.Enabled = True then begin
Timer1.Enabled := false;
Button1.Caption := 'Start';
end else begin
Timer1.Enabled := true;
Button1.Caption := 'Stop';
end;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
X := 0;
Y := 0;
```

```
Z := 0;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby w komponencie label był wyświetlony aktualny czas.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Timer1	TTimer
Label1	TLabel

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Sprawdzamy, czy komponent Timer1 ma właściwość Enabled ustawioną na True, oraz czy właściwość Interval równa się 1000

3) Do obsługi zdarzenia OnTimer wpisujemy poniższy kod:

```
Label1.Caption := TimeToStr(Time);    //kod pobiera z systemy aktalną
```

godzinę

4) Uruchamiamy program.

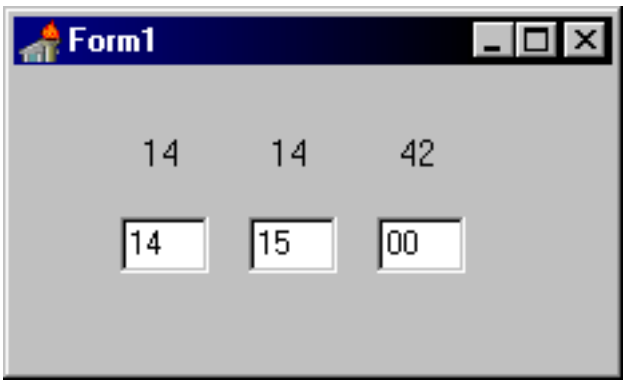
Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, ExtCtrls;

type
TForm1 = class(TForm)
Timer1: TTimer;
Label1: TLabel;
procedure Timer1Timer(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Timer1Timer(Sender: TObject);
begin
Label1.Caption := TimeToStr(Time);
end;

end.
```

Cel :

Nasz program ma przypominać budzik. Po wpisaniu godziny program sprawdza czy jeszcze nie nadeszła. W przypadku jeśli taka sytuacja zaistniała wyświetla komunikat.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Label1	TLabel
Label2	TLabel
Label3	TLabel
Edit1	TEdit
Edit2	TEdit
Edit3	TEdit

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Do obsługi funkcji OnTimer komponetu TTimer dodajemy kod:

```
var
A, B, C : String;
begin
A := FormatDateTime('hh', Time);
Label1.Caption := A;
B := FormatDateTime('nn', Time);
Label2.Caption := B;
C := FormatDateTime('ss', Time);
Label3.Caption := C;

if Edit3.Text <> '' then
if Edit2.Text <> '' then
if Edit1.Text <> '' then
if Edit3.Text = C then begin
if Edit2.Text = B then begin
if Edit1.Text = A then begin
MessageDlg('Nadeszła twoja godzina...', mtInformation , [mbOk], 0);
end;
end;
end;
```

3) Klikamy w pustym miejscu formulaarza i z zakładki Event wybieramy funkcję OnActive. Dodajemy kod:

```
Label1.Caption := FormatDateTime('hh', Time);
Label2.Caption := FormatDateTime('nn', Time);
Label3.Caption := FormatDateTime('ss', Time);
```

4) Uruchamiamy program.

Kod źródłowy :

```
unit Unit1;
```

```
interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls, StdCtrls;

type
TForm1 = class(TForm)
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Edit1: TEdit;
Timer1: TTimer;
Edit3: TEdit;
Edit2: TEdit;
procedure Timer1Timer(Sender: TObject);
procedure FormActivate(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Timer1Timer(Sender: TObject);
var
A, B, C : String;
begin
A := FormatDateTime('hh', Time);
Label1.Caption := A;
B := FormatDateTime('nn', Time);
Label2.Caption := B;
C := FormatDateTime('ss', Time);
```

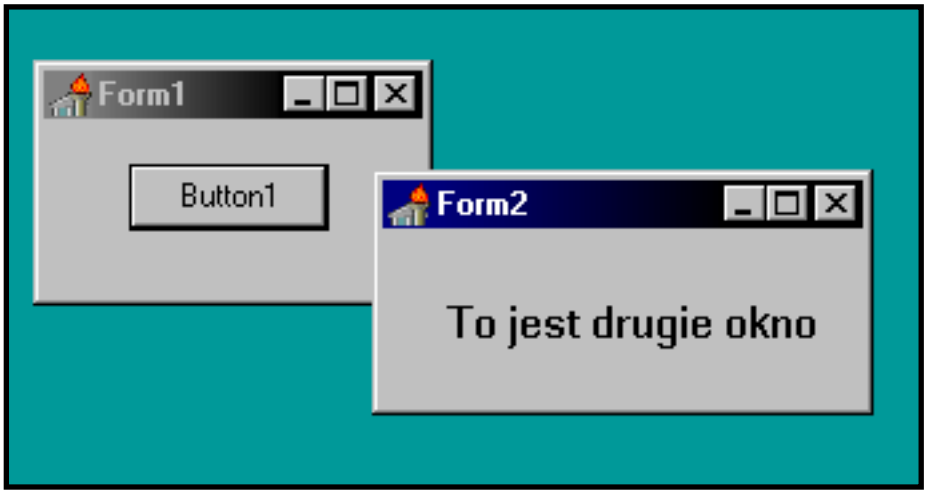
```
Label3.Caption := C;

if Edit3.Text <> '' then
if Edit2.Text <> '' then
if Edit1.Text <> '' then
if Edit3.Text = C then begin
if Edit2.Text = B then begin
if Edit1.Text = A then begin
MessageDlg('Nadeszła twoja godzina...', mtInformation , [mbOk], 0);
end;
end;
end;
end;

procedure TForm1.FormActivate(Sender: TObject);
begin
Label1.Caption := FormatDateTime('hh', Time);
Label2.Caption := FormatDateTime('nn', Time);
Label3.Caption := FormatDateTime('ss', Time);
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po kliknięciu na przycisk zostało otworzone nowe okno

---

Końcowy efekt :



Metoda :

1) Wybieramy z menu File|New Application

2) Wstawiamy przycisk

3) Z menu wybieramy File|New Form

4) Zmieniamy właściwość name formularza na - Form2.

5) Wstawiamy do nowo utworzonego okna komponent TLabel, zmieniamy jego właściwość Caption na - "To jest nowe okno".

6) Do obsługi funkcji OnClick wstawionego przycisku dodajemy kod:

Form2.ShowModal;

7) Uruchamiamy program, jeżeli ukarze się okno z pytaniem odpowiadamy

Yes i ponownie uruchamiamy program.

Kod źródłowy pierwszego formularza:

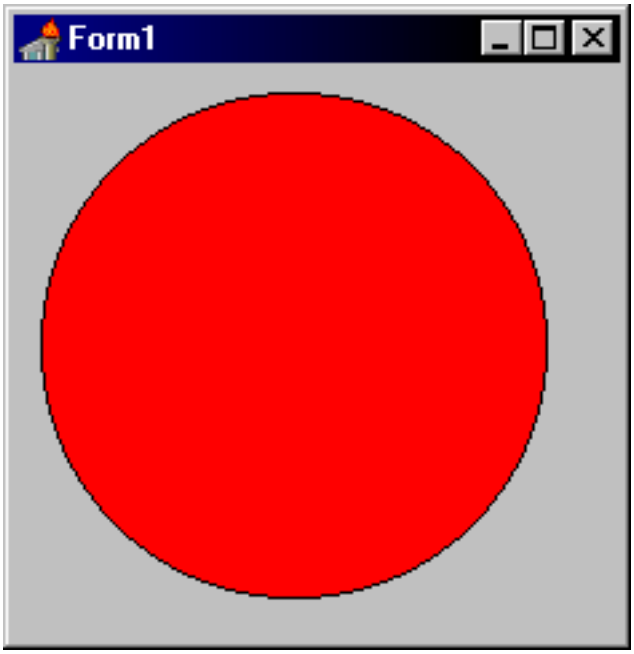
```
unit Unit1;

interface

uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Button1: TButton;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

uses Unit2;

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
Form2.ShowModal;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po uruchomieniu programu został wyświetlony okrąg

---

Końcowy efekt :



Metoda :

1) Tworzymy nowy projekt

2) Klikamy w pustym miejscu formularza i przechodzimy do zakładki Event. Wybieramy funkcję OnPaint

3) Wstawiamy poniższy kod:

```
Canvas.Brush.Color := clRed;
Canvas.Brush.Style := bsSolid;
Canvas.Ellipse(10,10, 200, 200);
```

4) Uruchamiamy program.

Porada pieerwsza :

W powyższym przykładzie wykonaliśmy rysunek elipsy, jednak płótno (Canvas) umożliwia jeszcze rysowanie paru innych figur. Oto one:

Rodzaj rysunku:	Przykładowy kod
Ellipsa	Canvas.Ellipse(50, 50, 200, 200);
Łuk	Canvas.Arc(0, 0, 100, 100, 0, 20, 90, 180);
Kwadrat	Canvas.Rectangle(20, 20, 100, 200);
Text	Canvas.TextOut(30, 30, 'text');
Linia	Canvas.MoveTo(0, 0); Canvas.LineTo(100, 100);

Porada druga :

Nasze płótno(Canvas) posiada dodatkowe opcje. Ustawienia dotyczące pędzla dostępne są poprzez atrybut Pen, który posiada m.in. takie ustawienia jak:

Opis:	Przykładowy kod
Zmienia kolor pędzla	Canvas.Pen.Color := clBlue;
Zmienia grubość pędzla	Canvas.Pen.Width := x

Drugim atrybutem dostępnym przez płótno jest Brush. Posiada m.in. ustawienia takie jak:

Opis:	Przykładowy kod
Zmienia color wypełnienia	Canvas.Brush.Color := clBlue;
Zmienia styl wypełnienia. Posiada takie wartości jak: bsSolid, bsCross, bsDiagCross, bsVertical, bsHorizontal.	Canvas.Brush.Style := bsSolid;

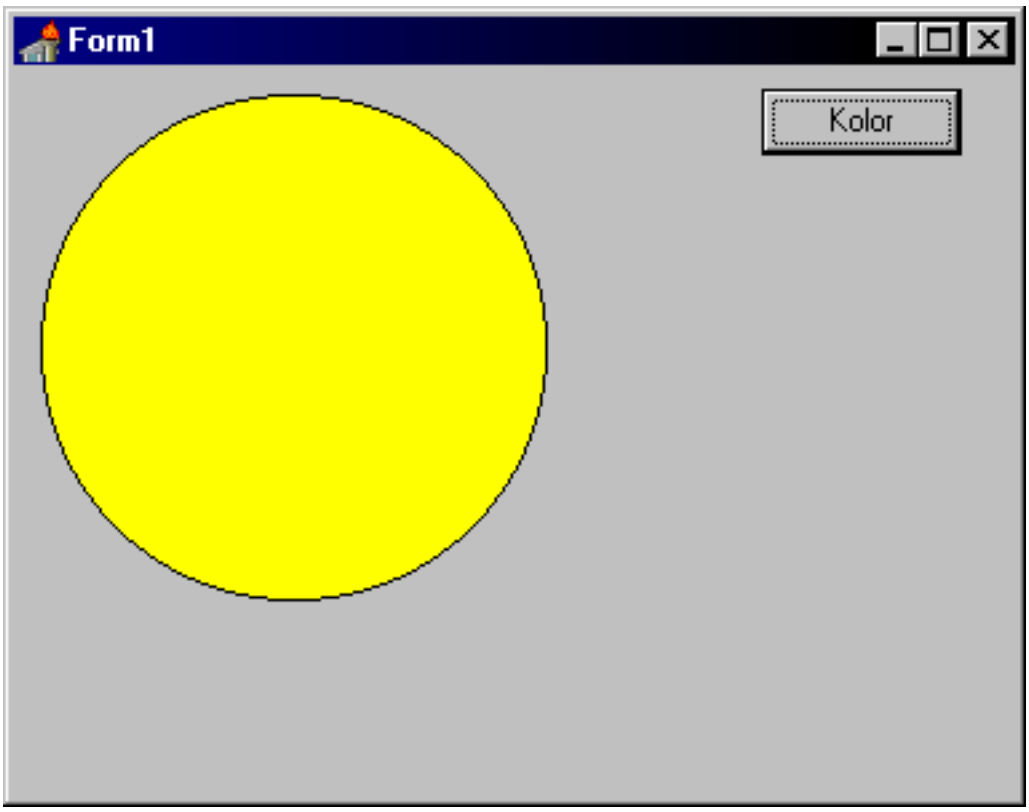
Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type
TForm1 = class(TForm)
procedure FormPaint(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormPaint(Sender: TObject);
begin
Canvas.Brush.Color := clRed;
Canvas.Brush.Style := bsSolid;
Canvas.Ellipse(10,10, 200, 200);
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po kliknięciu na przycisk zostało wyświetlone okno z kolorami, zaś po wybraniu koloru zostało narysowana elipsa o wybranym kolorze.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Button1	TButton
ColorDialog1	TColorDialog

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Do obsługi zdarzenia OnClick przycisku dodajemy kod:

```
ColorDialog1.Execute;
Canvas.Brush.Color := ColorDialog1.Color;
Canvas.Ellipse(10, 10, 200,200);
```

3) Uruchamiamy program.

Kod źródłowy :

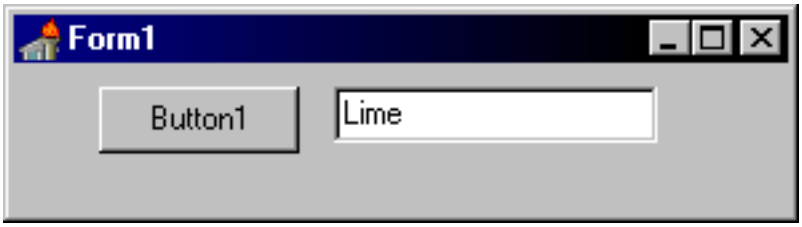
```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Button1: TButton;
ColorDialog1: TColorDialog;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation
```

```
{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
ColorDialog1.Execute;
Canvas.Brush.Color := ColorDialog1.Color;
Canvas.Ellipse(10, 10, 200,200);
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po wybraniu przez urzytkownika koloru z palety, nazwa koloru została wpisana do pola Edit.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Button1	TButton
ColorDialog1	TColorDialog
Edit1	TEdit

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Do obsługi zdarzenia OnClick przycisku dodajemy kod:

```
ColorDialog1.Execute;
Edit1.Text := IntToStr(ColorDialog1.Color);
if Edit1.Text = '0' then Edit1.Text := 'Black';
```

```
if Edit1.Text = '16711680' then Edit1.Text := 'Blue';
if Edit1.Text = '16776960' then Edit1.Text := 'Aqua';
if Edit1.Text = '65535' then Edit1.Text := 'Yellow';
if Edit1.Text = '255' then Edit1.Text := 'Red';
if Edit1.Text = '65280' then Edit1.Text := 'Lime';
if Edit1.Text = '32768' then Edit1.Text := 'Green';
if Edit1.Text = '16711935' then Edit1.Text := 'Fuchsia';
if Edit1.Text = '8388736' then Edit1.Text := 'Purple';
if Edit1.Text = '12632256' then Edit1.Text := 'Silver';
if Edit1.Text = '8421504' then Edit1.Text := 'Gray';
if Edit1.Text = '16777215' then Edit1.Text := 'White';
if Edit1.Text = '8421376' then Edit1.Text := 'Teal';
if Edit1.Text = '128' then Edit1.Text := 'Maroon';
```

3) Uruchamiamy program.

Porady :

Powyższy kod zapisuje kod liczbowy wybranej liczby do pola Edit, jeżeli numer został zdefiniowany przez jedną z dalszych metod zmienia nazwę na angielską. W przypadku jeżeli kolor nie został zdefiniowany wyświetla kod numerowy. Listę kolorów możemy powiększyć poprzez dodanie kodu:
```
if Edit1.Text = 'numer' then Edit1.Text := 'nazwa';
```

Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Button1: TButton;
ColorDialog1: TColorDialog;
```

```
Edit1: TEdit;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
begin
ColorDialog1.Execute;
Edit1.Text := IntToStr(ColorDialog1.Color);
if Edit1.Text = '0' then Edit1.Text := 'Black';
if Edit1.Text = '16711680' then Edit1.Text := 'Blue';
if Edit1.Text = '16776960' then Edit1.Text := 'Aqua';
if Edit1.Text = '65535' then Edit1.Text := 'Yellow';
if Edit1.Text = '255' then Edit1.Text := 'Red';
if Edit1.Text = '65280' then Edit1.Text := 'Lime';
if Edit1.Text = '32768' then Edit1.Text := 'Green';
if Edit1.Text = '16711935' then Edit1.Text := 'Fuchsia';
if Edit1.Text = '8388736' then Edit1.Text := 'Purple';
if Edit1.Text = '12632256' then Edit1.Text := 'Silver';
if Edit1.Text = '8421504' then Edit1.Text := 'Gray';
if Edit1.Text = '16777215' then Edit1.Text := 'White';
if Edit1.Text = '8421376' then Edit1.Text := 'Teal';
if Edit1.Text = '128' then Edit1.Text := 'Maroon';
end;

end.
```

Cel :

Chcemy uzyskać efekt prostej animacji. Bedzie ona wyświetlać kolejne obrazki Image1.bmp, Image2.bmp, Image3.bmp

---

Końcowy efekt :

Potrzebne komponenty :

Nazwa	Klasa
Image1	TImage
Timer1	TTimer

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Znajdz w okienku z kodem sekcję implementation i dodaj pod nią poniższą deklarację zmiennych:

```
var
numer : integer;
wgore : boolean;
```

3) Klikamy podwójnie na komponent Timer1. Dodajemy kod:

```
var
nazwa : string;
begin
if wgore = true then numer := numer + 1;
if wgore = false then numer := numer - 1;
nazwa := 'Image' + IntToStr(numer) + '.bmp';
Image1.Picture.Loadfromfile(nazwa);
if numer = 3 then wgore := false;
if numer = 1 then wgore := true;
```

4) klikamy podwójnie w pustym miejscu formularza i wpisujemy poniższy kod:

```
numer := 0;
wgore := true;
```

5) Uruchamiamy program.

Kod źródłowy :

```
unit Unit1;

interface

uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls;

type
TForm1 = class(TForm)
Image1: TImage;
Timer1: TTimer;
procedure Timer1Timer(Sender: TObject);
procedure FormCreate(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

var
numer : integer;
wgore : boolean;

procedure TForm1.Timer1Timer(Sender: TObject);
var
nazwa : string;
begin
if wgore = true then numer := numer + 1;
if wgore = false then numer := numer - 1;
nazwa := 'Image' + IntToStr(numer) + '.bmp';
Image1.Picture.Loadfromfile(nazwa);
if numer = 3 then wgore := false;
if numer = 1 then wgore := true;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
```

```
numer := 0;
wgore := true;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po najechaniu myszą na obrazek(Image1a.bmp) został wyświetlony drugi obrazek (Image1b.bmp). Ma to przypominać prosty Rollover.

Końcowy efekt :

Potrzebne komponenty :

Nazwa	Klasa
Image1	TImage

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Klikamy podwójnie na obrazku i wybieramy rysunek początkowy.

3) Znajdujemy sekcję implementation i dodajemy deklarację zminnych, zapobiegnie ona mryganiu obrazka na niektórych komponentach:

```
var
Image : boolean;
```

4) Klikamy podwójnie w pustym miejscu formularza i dodajemy kod:

```
Image := false;
```

5) Klikamy raz na obrazku i z zakładki Events wybieramy zdarzenie OnMouseMove. Dodajemy poniższy kod:

```
if Image = False then Image1.Picture.LoadFromFile('Image1b.bmp');
Image := true;
```

6) Klikamy w pustym miejscu formularza i ponownie z zakładki Evants wybieramy zdarzenie OnMouseMove. Tym razem dodajemy kod:

```
if Image = True then Image1.Picture.LoadFromFile('Image1a.bmp');
Image := False;
```

7) Uruchamiamy program.

Kod źródłowy :

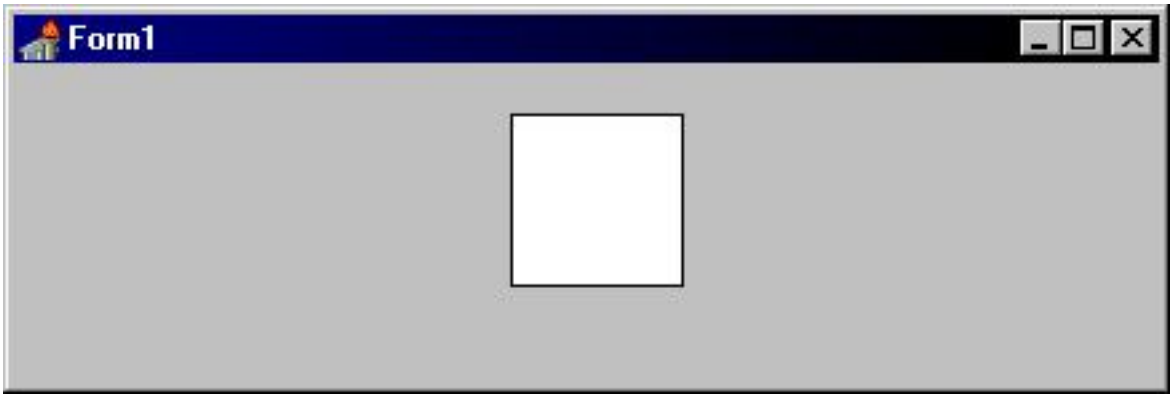
```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls;

type
TForm1 = class(TForm)
Image1: TImage;
procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
```

```
Y: Integer);
procedure FormCreate(Sender: TObject);
procedure FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation
var
Image : boolean;

{$R *.DFM}

procedure TForm1.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
begin
if Image = False then Image1.Picture.LoadFromFile('Image1b.bmp');
Image := true;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
Image := false;
end;

procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
begin
if Image = True then Image1.Picture.LoadFromFile('Image1a.bmp');
Image := False;
end;

end.
```

Cel :

Chcemy uzyskać efekt, aby po wciśnieciu jednej ze strzałek na klawiaturze komponent Shape został przesunienty o 10 pixeli w wybraną stronę.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Shape1	TShape

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Klikamy w pustym miejscu formularza i z zakładki Events wybieramy zdarzenie OnKeyDown. Wpisujemy poniższy kod:

```
if Key = VK_LEFT then Shape1.left := Shape1.left - 10;
if Key = VK_RIGHT then Shape1.left := Shape1.left + 10;
```

```
if Key = VK_UP then Shape1.top := Shape1.top - 10;
if Key = VK_DOWN then Shape1.top := Shape1.top + 10;
```

3) Uruchamiamy program.

Porady

W powyższym przykładzie poznałeś wirtualne kody klawiszy strzałek. Poniżej znajdziesz kody jeszcze kilku podstawowych klawiszy:

Kod klawisza:	Nazwa klawisza:
VK_RETURN	Enter
VK_SPACE	Spacja
VK_ESC	Esc
VK_SHIFT	Shift
VK_CONTROL	Ctrl
VK_MENU	Alt
VK_TAB	Tab
VK_BACK	Backspace
VK_INSERT	Insert
VK_HOME	Home
VK_PRIOR	Page Up
VK_DELETE	Delete
VK_END	End
VK_NEXT	Page Down
VK_0 ... VK_9	0 - 9
VK_NUMPAD0 ... VK_NUMPAD9	Numeryczne 0 - 9
VK_A ... VK_Z	Litery od A do Z

VK_F1 ... VK_F12	F1 - F12
VK_DIVIDE	Dzielenie
VK_MULTIPLY	Mnożenie
VK_SUBTRACT	Odejmowanie
VK_ADD	Dodawanie

Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
ExtCtrls;

type
TForm1 = class(TForm)
Shape1: TShape;
procedure FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word;
Shift: TShiftState);
begin
```

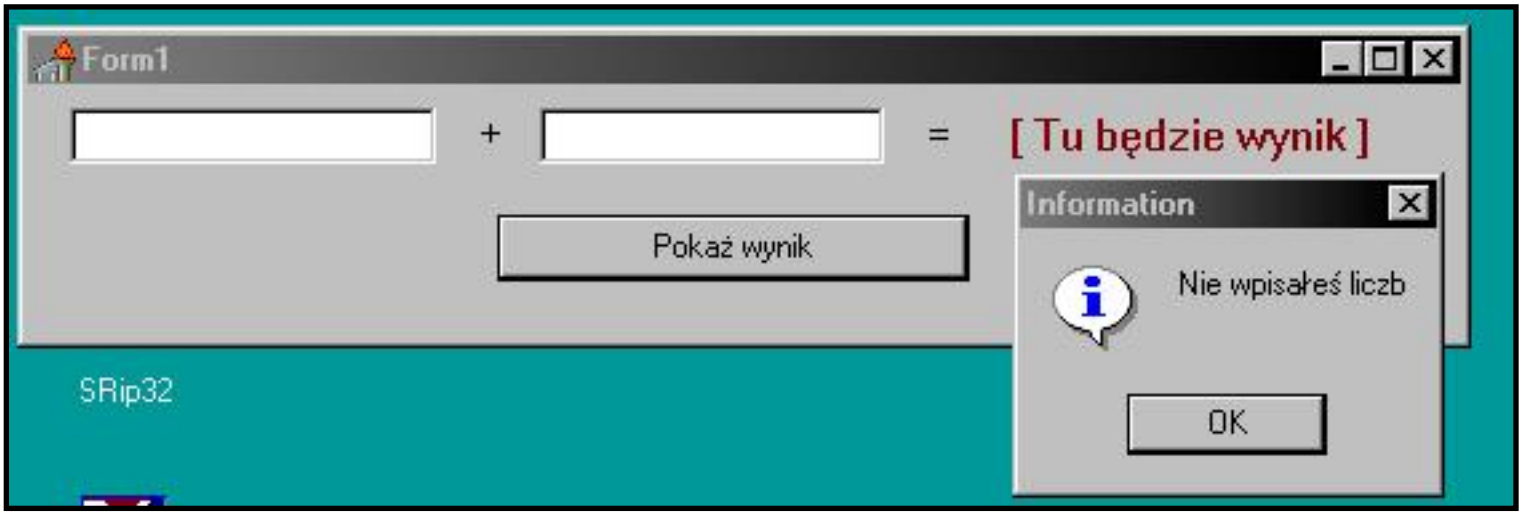
```
if Key = VK_LEFT then Shape1.left := Shape1.left - 10;
if Key = VK_RIGHT then Shape1.left := Shape1.left + 10;
if Key = VK_UP then Shape1.top := Shape1.top - 10;
if Key = VK_DOWN then Shape1.top := Shape1.top + 10;
end;

end.
```

Cel :

Poprawimy pracę naszego kalkulatora, a dokładniej obsłużymy wyjątek który powstawał, gdy użytkownik nie wpisał liczb w pola i kliknął przycisk

---

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Edit1	TEdit
Edit2	TEdit
Label1	TLabel
Label2	TLabel
Label3	TLabel
Button1	TButton

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa"

2) Zmieniamy właściwości Caption lub Text według własnego uznania lub sugerując się rysunkiem

3) Do obsługi funkcji OnClick przycisku dodajemy poniższy kod. Linie które zostały dodane do pierwszej wersji kalkulatora zostały dodatkowo pogrubione.

```
procedure TForm1.Button1Click(Sender: TObject);
var
X, Y, Z : Integer;
begin
try
X := StrToInt(Edit1.text);
Y := StrToInt(Edit2.text);
Z := X + Y;
Label3.Caption := IntToStr(Z);
except
MessageDlg('Nie wpisałeś liczb', mtInformation, [mbOk], 0);
end;
```

4) Uruchamiamy program.

Trochę teorii

Obejmując dane wyrażenia znacznikami **try ... except** możemy obsłużyć każdy wyjątek który może powsać. Pod słowem except wpisujemy co program ma zrobić po zaistnieniu wyjątku, gdy już to uczynimy zamykamy deklarację znacznikiem **end;**

Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Edit1: TEdit;
Label1: TLabel;
Edit2: TEdit;
Label2: TLabel;
Label3: TLabel;
Button1: TButton;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
X, Y, Z : Integer;
begin
try
X := StrToInt(Edit1.text);
Y := StrToInt(Edit2.text);
Z := X + Y;
Label3.Caption := IntToStr(Z);
except
```

```
MessageDlg('Nie wpisałeś liczb', mtInformation, [mbOk], 0);
end;
end;

end.
```

Cel :

Nasz program bedzie zliczał sumę liczb od 0 do liczby wskazanej przez urzytkownika np. 1 + 2 + 3 = 6

Końcowy efekt :

Potrzebne komponenty :

Nazwa	Klasa
Button1	TButton
Label1	TLabel
Label2	TLabel
Edit1	TEdit

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa".

2) Klikamy podwójnie na przycisk i dodajemy poniższy kod:

```
var
I, S, X : Integer;
begin
S := 0;
X := StrToInt(Edit1.text);   {Zminnej S przypisujemy wartość początkową 0 zaś, zmienna X otrzymuje wartość wpisaną przez użytkownika}
for I := 0 to X do begin   {funkja wskazuje ile razy ma zostać powtórzony poniższy kod}
S := S + I; {Do wartośći S dodajemy wartość I}
end;
Label2.Caption := IntToStr(S);
```

3) Uruchamiamy program.

Porady:

Pętle umożliwiają nam szybsze wykonanie wielu rzecz. Przykładowo chcielibyśmy, aby do zminnej **S** 10razy została dodana zminna **X**. Można to zapisać w postaci:

```
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
S := S + X;
```

lub

```
for I := 0 to 10 do begin S := S + X; end;
```

I co? Który sposób bardziej Ci odpowiada?
Powyższa pętla wykonuje 10 razy zdarzenie **S := S + X;**
Przy każdym kolejnym wykonaniu pętli wartość **I** wzrasta o 1. Gdy osiągnię

wartość wpisaną po słówku **to** kończy wykonywanie pętli i przechodzi do dalszych operacji.

Kod źródłowy :

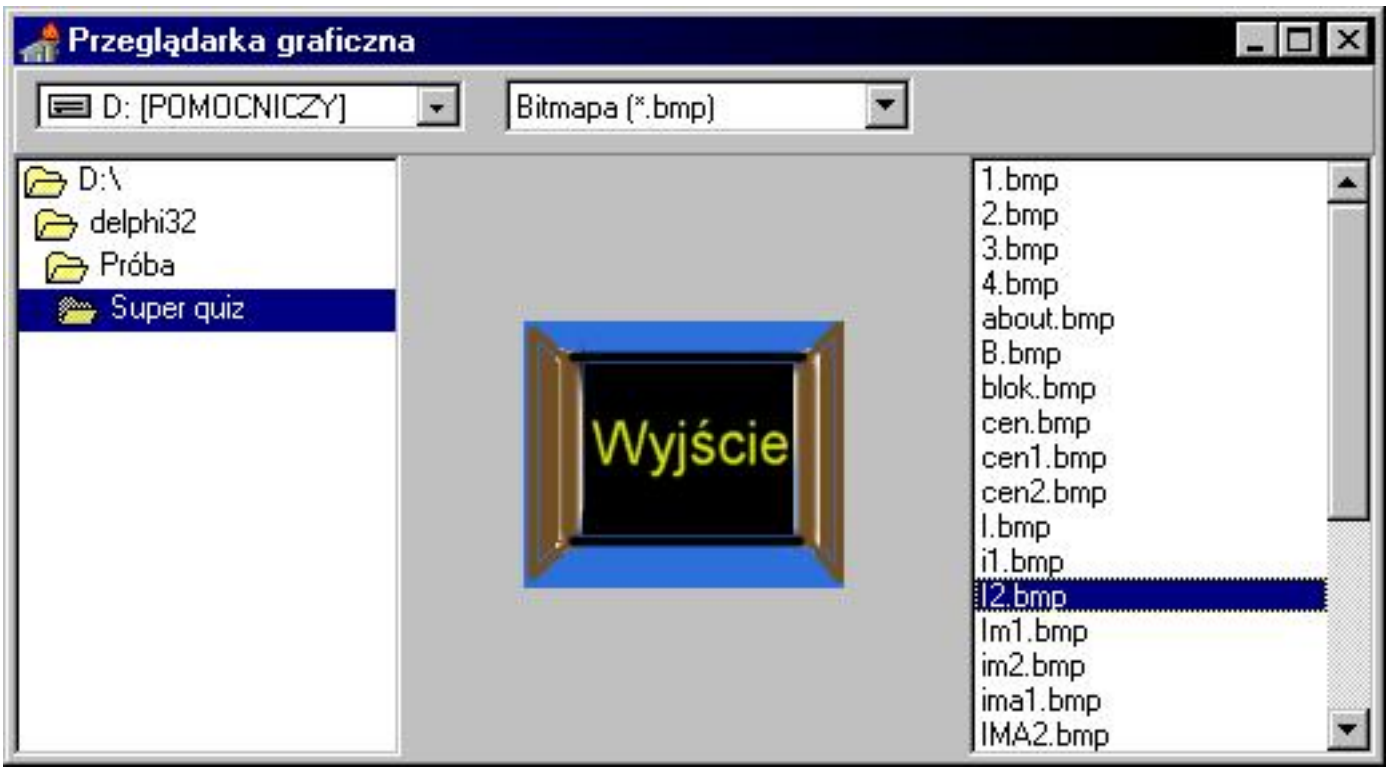
```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls;

type
TForm1 = class(TForm)
Edit1: TEdit;
Button1: TButton;
Label1: TLabel;
Label2: TLabel;
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Button1Click(Sender: TObject);
var
I, S, X : Integer;
begin
S := 0;
```

```
X := StrToInt(Edit1.text);
for I := 0 to X do begin
S := S + I;
end;
Label2.Caption := IntToStr(S);
end;
```

Cel :

Stworzymy prostą przeglądarkę graficzną opierającą się na drzewie katalogów. Po wybraniu pliku graficznego obrazek zostanie załadowany do komponentu TImage.

Końcowy efekt :



Potrzebne komponenty :

Nazwa	Klasa
Panel1	TPanel
DriveComboBox1	TDriveComboBox
FilterComboBox1	TFilterComboBox
DirectoryListBox1	TDirectoryListBox

FileListBox1	TFileListBox
Image1	TImage

Metoda :

1) Wstawiamy komponenty, wypisane w powyższej tabeli i zmieniamy im właściwość name na taką jaka jest w kolumnie "Nazwa". Komponenty TDriveComboBox, FilterComboBox1 wstawiamy na Panel

2) Właściwość DirList komponentu DriveComboBox1 ustawiamy na DirectoryListBox1

3) Właściwość FileList komponentu DirectoryListBox1 ustawiamy na FileListBox1

4) Właściwość FileList komponentu FilterComboBox1 zmieniamy na FileListBox1

5) Klkamy raz na komp. FilterComboBox1 i w Object Inspector podwójnie klikamy przy właściwości Filter. W otworzonym okienku w kolumnie " Filter Name " wpisujemy - Bitmapa (*.bmp), zaś w kolumnie " Filter " - *.bmp

6) Klikamy podwójnie na komponencie - TFileListBox i w wygenerowanym zdarzeniu wpisujemy poniższy kod:

```
var
x:string;
begin
x:= filelistbox1.FileName;
if x = '' then Exit;
Image1.Picture.Loadfromfile(x);
```

7) Właściwości komponentu TImage zmień następująco:
AutoSize = true
Align = alClient
Center = true

8) Uruchamiamy program.

Kod źródłowy :

```
unit Unit1;

interface

uses
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
StdCtrls, FileCtrl, ExtCtrls;

type
TForm1 = class(TForm)
FileListBox1: TFileListBox;
DirectoryListBox1: TDirectoryListBox;
Image1: TImage;
Panel1: TPanel;
DriveComboBox1: TDriveComboBox;
FilterComboBox1: TFilterComboBox;
procedure FileListBox1Change(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;

var
Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.FileListBox1Change(Sender: TObject);
var
x:string;
begin
```

```
x:= filelistbox1.FileName;
if x = '' then Exit;
Image1.Picture.Loadfromfile(x);
end;


end.
```

Paleta kolorów

W tej lekcji zajmiemy się zabawą na palecie kolorów. Korzysta się z niej bardzo często tworząc ciekawe efekty graficzne lub np. gradient tła. Możliwe, że w tej chwili myślisz sobie jaka trudność napisać coś takiego:

Form1.Color := clBlue;

A czy wiesz co oznacza taki kod:

Form1.Color := RGB(255, 0, 0);

Jeżeli tak i wiesz jak manipulować tymi cyframi nie masz po co czytać dalszej części tego kursu. Jak już pewnie zauważyłeś dany kolor można zadeklarować poprzez poszczególne składniki, którymi są kolory czerwony, zielony, niebieski. I tak np. żeby zadeklarować kolor czerwony trzeba napisać coś takiego RGB(255, 0, 0) kolor czarny - RGB(0, 0, 0). Pewnie w tej chwili zastanawiasz się po co się męczyć z deklarowaniem kolorów w takiej postaci. Może to być przydatne wtedy kiedy zadeklarujemy jakiś kolor i chcemy go stopniewo rozjaśniać. Pozostaje tylko jedno pytanie? Jednak skąd teraz dowiedzieć się jakiemu kolorowi odpowiadają jakie współczynniki. Wystarczy otworzyć dowolną paletę Windows. Gdzieś na niej muszą się znajdować takie pola jak: czerwony, zielony, niebieski.

No dobrze, ale teraz trochę praktyki. Wstaw do nowo utworzonego formularza komponent TTimer dostępny na palecie System. Pod sekcją implementation dodaj kod:

```
var
czerwony, zielony, niebieski : Integer;
```

Kliknij dwa razy na tle formularza, aby wygenerować zdarzenie OnCreate. Wpisz:

```
czerwony := 255;
zielony := 0;
niebieski := 0;
```

Kliknij podwójnie na wcześniej wstawionym komponencie. Dodaj kod:

```
czerwony := czerwony - 10;
if czerwony < 10 then begin
czerwony := 255;
```

```
end;
Form1.Color := RGB(czerwony, zielony, niebieski);
```

Teraz po uruchomieniu aplikacji tło aplikacji będzie stopniowo ciemnieć od koloru czerwonego do czarnego. Fajnie wygląda co?