

Przemysław Pisula

ZŁOTE
MYŚLI

Delphi

**31 przydatnych
programów**

**Naucz się pełniej wykorzystywać
możliwości Delphi tworząc
przykładowe programy**

Niniejszy **darmowy** ebook zawiera fragment
pełnej wersji pod tytułem:

“Delphi – 31 przydatnych programów”

Aby przeczytać informacje o pełnej wersji, [kliknij tutaj](#)

Darmowa publikacja dostarczona przez
www.twojebook.net

Niniejsza publikacja może być kopiowana, oraz dowolnie rozprowadzana tylko i wyłącznie w formie dostarczonej przez Wydawcę. Zabronione są jakiegokolwiek zmiany w zawartości publikacji bez pisemnej zgody wydawcy. Zabrania się jej odsprzedaży, zgodnie z [regulaminem Wydawnictwa Złote Myśli](#).

© Copyright for Polish edition by ZloteMysli.pl

Data: 07.02.2007

Tytuł: Delphi – 31 przydatnych programów (fragment utworu)

Autor: Przemysław Pisula

Projekt okładki: Marzena Osuchowicz

Korekta: Sylwia Fortuna

Skład: Anna Popis-Witkowska

Internetowe Wydawnictwo Złote Myśli

Netina Sp. z o. o.

ul. Daszyńskiego 5

44-100 Gliwice

WWW: www.ZloteMysli.pl

EMAIL: kontakt@zlotemysli.pl

Wszelkie prawa zastrzeżone.

All rights reserved.

SPIS TREŚCI

<u>WSTĘP</u>	5
<u>PROGRAM „MEMO, EDIT, LISTA, COMBO”</u>	8
<u>PROGRAM „ANKIETA”</u>	14
<u>PROGRAM „KSIĄŻKA ADRESOWA”</u>	28
<u>PROGRAM „MAINMENU I POPUPMENU”</u>	35
<u>PROGRAM „PRZEGLĄDARKA GRAFIKI I SLAJDÓW”</u>	47
<u>PROGRAM „KOLORY FORMATKI”</u>	60
<u>PROGRAM „ODCZYT KODÓW I NUMERÓW KŁAWISZY”</u>	66
<u>PROGRAM „KALKULATOR”</u>	70
<u>PROGRAM „ZGADUJ-ZGADULA”</u>	89
<u>PROGRAM „ODTWARZACZ MUZYCZNY”</u>	100
<u>WŁASNY KOMPONENT „TDZIAŁANIA”</u>	110
<u>PROGRAM WYKORZYSTUJĄCY KOMPONENT TDZIAŁANIA</u>	116
<u>PROGRAM „PRZECIAGNIJ I UPUŚĆ”</u>	121
<u>PROGRAM „KOLORY RGB”</u>	126
<u>PROGRAM „TWORZENIE OBIEKTÓW”</u>	131
<u>PROGRAM „RÓWNANIE KWADRATOWE – OBIEKTOWO”</u>	140
<u>PROGRAM „AKTYWNA FORMA”</u>	150
<u>PROGRAM „BAZA DANYCH FIRMY”</u>	155
<u>PROGRAM „TWORZENIE, KOPIOWANIE, USUWANIE”</u>	170
<u>PROGRAM „RYSOWANIE SAMOCZYNNE”</u>	175
<u>PROGRAM „BAZA DANYCH”</u>	180
<u>PROGRAM „BUDUJEMY INTRO PROGRAMU”</u>	196
<u>PROGRAM „ZMIENŃ TAPETĘ, OTWÓRZ INNY PROGRAM”</u>	200
<u>PROGRAM „ODTWARZACZ FILMÓW”</u>	205
<u>PROGRAM „ARKUSZ KALKULACYJNY”</u>	213
<u>PROGRAM „GIEŁDA SAMOCHODOWA”</u>	221
<u>PROGRAM „KOMUNIKATOR SIECIOWY”</u>	231
Program „Klient”.....	231
Program „Server”.....	236
<u>PROGRAM „PRZEGLĄDARKA STRON WWW”</u>	242
<u>PROGRAM „EDYTOR TEKSTU”</u>	254
<u>PROGRAM „EDYTOR HTML”</u>	272
<u>PROGRAM „RYSOWANIE MYSZKĄ”</u>	277

Wstęp

Niniejsza publikacja przedstawia 30 programów stworzonych w popularnym środowisku programistycznym, jakim jest **Delphi**. Przeznaczona jest dla początkujących, ale i takich, którzy mają już pewne doświadczenie w programowaniu.

Publikacja zakłada, że Czytelnik zapoznał się z podstawami **Object Pascal** oraz umie poruszać się w środowisku **Delphi**. Dlatego nie będę omawiał tutaj tych spraw.

Opis tych rzeczy można znaleźć w wielu książkach i podręcznikach.

Ebook ten ma na celu przedstawić Czytelnikowi praktyczne zastosowanie posiadanej wiedzy.

Książka przedstawia programy wraz z ilustracjami, kodem źródłowym i komentarzami.

Publikowane tutaj programy mogą stanowić wstęp do dalszej ich rozbudowy.

Na początku każdego programu Czytelnik znajdzie wyjaśnienie jego działania, opis zastosowanych komponentów, a dalej szczegółowe wyjaśnienie działania wszystkich użytych instrukcji, funkcji i procedur.

Znajduje się tutaj również sposób tworzenia własnego komponentu, aplikacji internetowych i zasady programowania obiektowego.

Sprawy podstawowe:

1. Do komponentu w kodzie programu odwołujemy się poprzez jego właściwość **Name**.

Dlatego jeśli ją zmienisz z domyślnej nadanej przez **Delphi** na swoją to pamiętaj, abyś jej konsekwentnie używał do końca.

Jeśli odwołujesz się do komponentu znajdującego się w innej formie, ale w tym samym programie, to jego nazwę **Name** poprzedzaj nazwą tego formularza.

2. Właściwości komponentów dostępne na zakładce **Properties** można zmieniać w czasie projektowania programu lub w czasie jego działania (w kodzie programu). W czasie projektowania po prawej stronie od nazwy wpisujesz jej wartość lub klikasz w strzałkę i wybierasz ją z listy. Przy niektórych z nich jest prostokąt z trzema kropkami, więc klikasz na niego i rozwija się edytor, w którym wpisujesz wartości.

3. Zdarzenia dla danego komponentu generujemy po przejściu na zakładkę **Events**. Zaznaczasz tam pole w kolumnie po prawej stronie od nazwy danego zdarzenia, jego kolor zmieni się na biały, a następnie klikasz w nie dwa razy. Nazwy procedur Delphi generuje automatycznie. Pamiętaj, abyś nic w nich nie zmieniał! Twoim zadaniem jest wpisanie instrukcji między **Begin** a **End** i ewentualne zadeklarowanie stałych zmiennych, tablic itp. nad słowem **Begin**, a przed **Procedure Nazwa (parametry)**.

4. W niniejszej publikacji, aby nie powtarzać dwa razy tego samego, nie będę przedstawiał treści procedur osobno dla każdego komponentu, tylko ujmę je wszystkie razem podając cały kod źródłowy modułu lub modułów, jeśli będzie ich kilka.

Obok każdej procedury znajdować się będzie opis, co to za procedura i do jakiego komponentu się odnosi. Wyjaśnienia i komentarze podane będą na końcu lub z boku funkcji lub procedury.

5. Jeśli chcesz używać polskich liter w **Delphi**, to ściągnij plik „**Polska klawiatura**” ze strony <http://www.borland.pl> i zainstaluj go na swoim komputerze. Zwróć tylko uwagę na numerację, abyś zainstalował plik przeznaczony do odpowiedniej wersji.

6. Jak zdobyć Delphi?

- PC WORLD KOMPUTER 10AB/2001– Delphi 6.0 Enterprise Trial
- PC WORLD KOMPUTER 11AB/2001– Delphi 5 Standard (Komercyjna)
- KOMPUTER ŚWIAT Ekspert – Delphi 7 Personal Edition
- KOMPUTER ŚWIAT Ekspert – Delphi 2005 Personal

Kody odblokowujące do wszystkich wersji otrzymujemy po bezpłatnym zarejestrowaniu się na stronie firmy BORLAND <http://www.borland.pl>. W przypadku programów: Baza danych i Komunikator sieciowy musisz dysponować wersją **Enterprise**.

Uwaga !: W publikacji dla większej przejrzystości użyto spacji między apostrofem a literą, wyrazem, nawiasem lub cyfrą. W pisanych przez siebie programach nie należy tego robić, ponieważ może to się przyczynić do nieprawidłowego działania programów.

W kodach źródłowych dołączonych do książki zastosowano prawidłowe kodowanie.

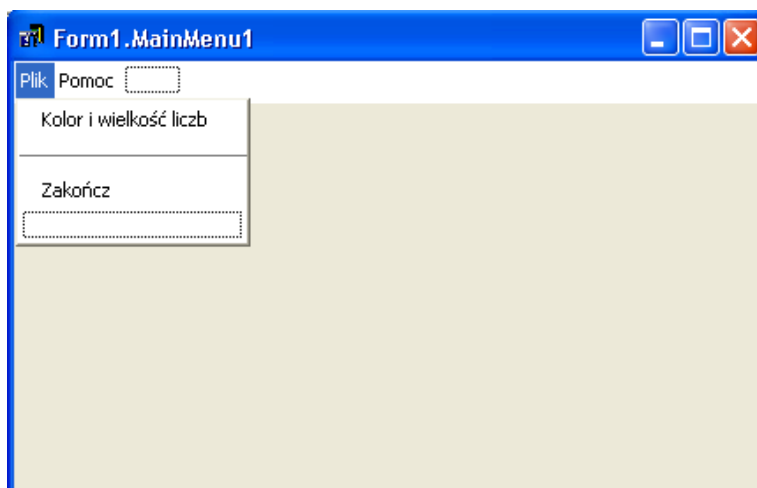
Program „Kalkulator”

Program jest typowym kalkulatorem wykonujący podstawowe działania arytmetyczne.

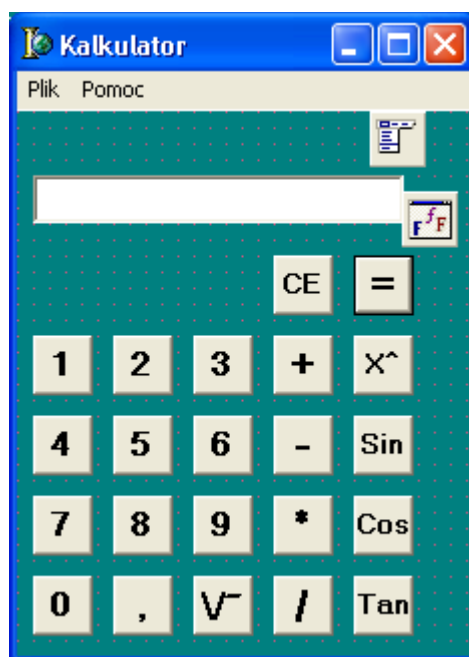
Wstaw na formę:

- 22 **Buttony**. Zmień wysokość (**Height**) i szerokość (**Width**) wszystkich na 30 pikseli. Możesz to zrobić za jednym zamachem obrysowując lub zaznaczając je wszystkie z wciśniętym klawiszem **Shift**. Zaznaczenie będzie wówczas miało kolor szary. Wówczas zmień w/w wartości. Pozmieniaj ich **Caption** jak na rysunku. Pierwiastek zrobiłem z ukośnika i znaku tyldy.
- jedno pole **Edit**. Wykasuj jego właściwość **Text**.
- **FontDialog (Dialogs)**
- **MainMenu (Standard)** Kliknij dwa razy w **MainMenu** i wpisz kolejno do **Caption** zaznaczając uprzednio poszczególne pola.
- „**Plik**”, „**Kolor i wielkość liczb**” i „**Zakończ**”. Zaznacz pole na prawo od **Plik** i wpisz kolejno do **Caption**:
- „**Pomoc**” i „**O programie**”.

Rys 19. MainMenu.



Rys. 20 Wygląd formy.



- dodamy drugą formę z informacją o programie.
- Wybierz **File** -> **New** -> **Other**. Zaznacz zakładkę Kalkulator i wybierz **AboutBox**.

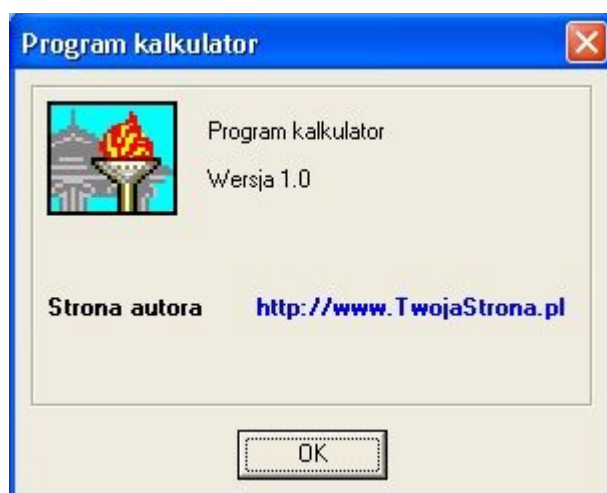
Jest to okienko dialogowe z przyciskiem **OK**. Pozmieniaj **Caption** umieszczonych tam **Labeli**, do **Image** załaduj jakiś obraz (właściwość **Picture**). W jednym Labelu wpisz adres swojej strony WWW. Będzie się ona otwierała po naciśnięciu na ten **Label**. Oczywiście zrobimy tak, aby po najechaniu na niego myszką kursor zmieniał kształt na „łapkę”

W **Label4** z adresem strony zmień kolor czcionki na niebieski (właściwość **Font**).

Wykorzystaliśmy gotowy szablon, ale nic nie stoi na przeszkodzie, aby takie okno zbudować ze zwykłej formatki. Należy wówczas jej właściwość **BorderStyle** ustawić na **bsDialog**.

Powinno to wyglądać mniej więcej tak, jak na rysunku.

Rys. 21 Wygląd formy drugiej „O programie”



Teraz należy dodać tę drugą formę do głównej. Zaznacz pierwszą (główną) formę i wybierz **File -> Use Unit**. Kiedy otworzy się okienko dodawania modułu, kliknij OK. **Delphi** doda drugą formę do programu i dokona odpowiedniego wpisu do modułu głównego. Nie będziesz tego musiał wpisywać ręcznie.

Najpierw oprogramujemy zdarzenia dla drugiej formy.

Kod źródłowy programu „Kalkulator” modułu drugiego (AboutBox)

```
unit Unit3;
```

```
interface
```

```
uses Windows, SysUtils, Classes, Graphics, Forms, Controls,  
StdCtrls,
```

```
Buttons, ExtCtrls, ShellApi, jpeg;
```

```
// dodaj moduł ShellAPI
```

```
type
```

```
TAboutBox = class(TForm)
```

```
OKButton: TButton;
```

```
Image1: TImage;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Label3: TLabel;
```

```
Label4: TLabel;
```

```
        procedure Label4MouseDown(Sender: TObject; Button:
TMouseButton;
        Shift: TShiftState; X, Y: Integer);
        procedure Label4MouseMove(Sender: TObject; Shift: TShiftState;
X,
        Y: Integer);

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

var
    AboutBox: TAboutBox;

implementation

{$R *.dfm}

procedure TAboutBox.Label4MouseDown(Sender: TObject; Button:
TMouseButton;
    Shift: TShiftState; X, Y: Integer);
// onMouseDown dla Label4
begin
    ShellExecute( handle, ' open ', ' http://www.TwojaStrona.pl ', nil,
nil, SW_SHOW );
```

end;

(*

- *onMouseDown* zachodzi, kiedy znajdując się nad komponentem naciskamy lewy przycisk myszy.
- funkcja *ShellExecute* otwiera domyślną przeglądarkę z podanym adresem strony.

Handle – to uchwyt okna, *SW_SHOW* – oznacza wyświetl.

*Inna wartość to SW_HIDE ukryj. *)*

```
procedure TAboutBox.Label4MouseMove(Sender: TObject; Shift: TShiftState; X,
```

```
Y: Integer);
```

```
// onMouseMove dla Label4
```

```
begin
```

```
Label4.Cursor:=crHandPoint;
```

```
end;
```

```
end.
```

(* *onMouseMove* zachodzi, kiedy przesuwamy wskaźnik myszy nad komponentem.

*Tutaj podczas przesuwania wskaźnika myszy nad Labelem z adresem strony WWW zmieniamy kursor na „łapkę” crHandPoint *)*

Dla umieszczonego tam przycisku **OK** nie musisz wpisywać kodu. Jest już wpisany.

Wykorzystaliśmy funkcję **ShellExecute**, więc do listy **Uses AboutBox** dodaj moduł **ShellApi**.

Kod źródłowy programu „Kalkulator”. Moduł główny

W sekcji **Private** umieść nagłówki dwóch procedur pomocniczych

- **działanie**
- **fokus**

Pod słowem **implementation** umieścimy kilka zmiennych, by były widziane w całym module

```
unit Unit1;
```

```
interface
```

```
uses
```

```
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
    Controls, Forms,
```

```
    Dialogs, Buttons, StdCtrls, Menus;
```

```
type
```

```
    TForm1 = class(TForm)
```

```
        Edit1: TEdit;
```

```
Button1: TButton;  
Button2: TButton;  
Button3: TButton;  
Button4: TButton;  
Button5: TButton;  
Button6: TButton;  
Button7: TButton;  
Button8: TButton;  
Button9: TButton;  
Button10: TButton;  
Button11: TButton;  
Button12: TButton;  
Button13: TButton;  
Button14: TButton;  
Button15: TButton;  
Button16: TButton;  
Button17: TButton;  
MainMenu1: TMainMenu;  
Plik1: TMenuItem;  
Kolorliczb1: TMenuItem;  
N1: TMenuItem;  
Zakocz1: TMenuItem;  
Pomoc1: TMenuItem;  
Oprogramie1: TMenuItem;  
Button18: TButton;  
FontDialog1: TFontDialog;
```

Button19: TButton;

Button20: TButton;

Button21: TButton;

Button22: TButton;

procedure Button2Click(Sender: TObject);

procedure Zakocz1Click(Sender: TObject);

procedure Kolorliczb1Click(Sender: TObject);

procedure Button3Click(Sender: TObject);

procedure Button12Click(Sender: TObject);

procedure Button11Click(Sender: TObject);

procedure Button4Click(Sender: TObject);

procedure Button13Click(Sender: TObject);

procedure Button10Click(Sender: TObject);

procedure Button5Click(Sender: TObject);

procedure Button6Click(Sender: TObject);

procedure Button9Click(Sender: TObject);

procedure Button7Click(Sender: TObject);

procedure Button8Click(Sender: TObject);

procedure Button18Click(Sender: TObject);

procedure Button14Click(Sender: TObject);

procedure Button15Click(Sender: TObject);

procedure Button16Click(Sender: TObject);

procedure Button19Click(Sender: TObject);

procedure Button1Click(Sender: TObject);

procedure Button20Click(Sender: TObject);

```
procedure Button17Click(Sender: TObject);
procedure Oprogramie1Click(Sender: TObject);
procedure Button21Click(Sender: TObject);
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure Button22Click(Sender: TObject);

private
  { Private declarations }
  procedure dzialanie;
  // nagłówki procedur dzialanie i fokus
  procedure fokus;
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

uses Unit3;

(* tutaj Delphi dodał drugi modul. U mnie to akurat Unit3. To
samo można wpisać ręcznie *)

{$R *.dfm}
Var x, a, b: Currency;
```

```
// deklaracja zmiennych widzianych przez cały modul
znak: Char;
liczba: Boolean;

procedure TForm1.dzialanie;
begin
Case znak of
' + ': Edit1.Text:=FloatToStr(x + StrToFloat(Edit1.Text));
' - ': Edit1.Text:=FloatToStr(x - StrToFloat(Edit1.Text));
' * ': Edit1.Text:=FloatToStr(x * StrToFloat(Edit1.Text));
' / ': try Edit1.Text:=FloatToStr(x / StrToFloat(Edit1.Text));
      except on EZeroDivide do
// dzielenie z obsługą błędu
      ShowMessage(' Błąd dzielenia przez zero ! ');
      end;
' ^ ': begin a:=StrToInt(Edit1.Text);
// potęgowanie
      b:=Exp(a * Ln(x));
      Edit1.Text:=FloatToStr(b);
      end;
' s ': Edit1.Text:=FloatToStrF(Sin((Pi * x) / 180), ffGeneral, 3, 1);
// sinus
' c ': Edit1.Text:=FloatToStrF(Cos((Pi * x) / 180), ffGeneral, 3, 1);
// cosinus
' p ': Edit1.Text:=FloatToStr(Sqrt(x));
// pierwiastek drugiego stopnia
```

```
' t ': Edit1.Text:=FloatToStrF(ArcTan((Pi * x) / 180), ffGeneral, 3, 1);  
// tangens  
end;  
end;
```

(*

- procedura “działanie” wykrywa znak i odpowiednio do tego wykonuje przypisane do tego znaku działanie matematyczne.
 - *F*loatTo*S*tr*F*(Liczba, ffGeneral, Precyzja, X, Y) formatuje wyświetlaną liczbę
 - X – dokładność wyświetlania w formacie dziesiętnym
 - Y – minimalna liczba cyfr dla notacji wykładniczej
 - działanie *Exp*(a * *Ln*(x)) – podnosi liczbę „X” do dowolnej potęgi „a”
 - w przypadku dzielenia zastosowano obsługę błędu *Try...Except*, na wypadek gdyby ktoś chciał dzielić przez zero
 - w przypadku funkcji trygonometrycznych zapis *Sin*(Pi * X) / 180 jest przeliczeniem radianów na stopnie, ponieważ w Delphi wartość argumentu do tych funkcji przekazuje się w radianach.
- *)

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
// onClick dla przycisku “CE”
```

```
begin
```

```
Edit1.Text:=' ';
```

```
// wyczyść Edita
```

```
Edit1.SetFocus;
```

```
// ustaw wskaźnik w Edit
```

```
end;
```

```
(* Przycisk "CE" czyści pole Edit i ustawia w nim wskaźnik *)
```

```
procedure TForm1.Zakocz1Click(Sender: TObject);
```

```
// onClick dla "Zakończ" (w MainMenu)
```

```
begin
```

```
Close;
```

```
end;
```

```
procedure TForm1.Kolorliczb1Click(Sender: TObject);
```

```
(* onClick dla przycisku „Kolor i wielkość liczb” w MainMenu *)
```

```
begin
```

```
if FontDialog1.Execute Then
```

```
// jeśli otwarte jest okno wyboru czcionki
```

```
Edit1.Font:=FontDialog1.Font;
```

```
// to wybrane atrybuty czcionki przypisz do Edit1
```

```
end;
```

```
procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
```

```
fokus;
```

```
Edit1.Text:=Edit1.Text + ' 1 ';
```

```
end;
```

```
(*
```

```
- wywołaj procedurę fokus. Opis dalej.
```

- do pola *Edit1* dodaj to, co w nim wpisane plus znak naciśniętego *Buttona*. Czyli dodaje cyfry od 0...9 i przecinek.

*Procedury dla wszystkich przycisków z cyframi i przecinkiem są identyczne, zmienia się tylko cyfra i przecinek. Więc można je właściwie przekopiować zmieniając wartość na końcu. *).*

```
procedure TForm1.Button12Click(Sender: TObject);  
begin  
fokus;  
Edit1.Text:=Edit1.Text + ' 2 ';  
// podobnie j/w  
end;
```

```
procedure TForm1.Button11Click(Sender: TObject);  
begin  
fokus;  
Edit1.Text:=Edit1.Text + ' 3 ';  
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);  
begin  
fokus;  
Edit1.Text:=Edit1.Text + ' 4 ';  
end;
```

```
procedure TForm1.Button13Click(Sender: TObject);
```

```
begin
fokus;
  Edit1.Text:=Edit1.Text + ' 5 ';
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' 6 ';
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' 7 ';
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' 8 ';
end;

procedure TForm1.Button9Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' 9 ';
end;
```

```
procedure TForm1.Button7Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' o ';
end;
```

```
procedure TForm1.Button8Click(Sender: TObject);
begin
fokus;
  Edit1.Text:=Edit1.Text + ' , ' ;
end;
```

```
procedure TForm1.Button18Click(Sender: TObject);
// onClick dla przycisku “+”
begin
  x:=StrToFloat(Edit1.Text);
znak:= ' + ' ;
liczba:=False;
Edit1.Text:= ' ' ;
Edit1.SetFocus;
end;
```

(*

- do zmiennej “X” przypisz aktualną wartość z Edit1
- wykryty „znak: to „+”
- ustaw zmienną „liczba” na False

- *wyczyść Edita*
- *ustaw w Edit kursor*
- *dla pozostałych procedur wykonujących działania instrukcję są podobne, zmienia się tylko znak *)*

```
procedure TForm1.Button14Click(Sender: TObject);
```

```
begin
```

```
  x:=StrToFloat(Edit1.Text);
```

```
znak:=' - ';
```

```
liczba:=False;
```

```
Edit1.Text:=' ';
```

```
  Edit1.SetFocus;
```

```
end;
```

```
procedure TForm1.Button15Click(Sender: TObject);
```

```
begin
```

```
  x:=StrToFloat(Edit1.Text);
```

```
znak:=' * ';
```

```
liczba:=False;
```

```
Edit1.Text:=' ';
```

```
  Edit1.SetFocus;
```

```
end;
```

```
procedure TForm1.Button16Click(Sender: TObject);
```

```
begin
```

```
  x:=StrToFloat(Edit1.Text);
```

```
znak:=' / ';
```

Przemysław Pisula

```
liczba:=False;
Edit1.Text:=' ';
    Edit1.SetFocus;
end;
procedure TForm1.Button19Click(Sender: TObject);
begin
    x:=StrToFloat(Edit1.Text);
znak:=' ^ ';
liczba:=False;
Edit1.Text:=' ';
    Edit1.SetFocus;
end;

procedure TForm1.Button1Click(Sender: TObject);
// onClick dla “ = ”
begin
    dzialanie;
znak:=' = ';
liczba:=False;
end;

(*
- wykonaj odpowiednie dzialanie (procedura “dzialanie”)
- wykryty „znak” to „=”
- ustaw zmienną „liczba” na False *)

procedure TForm1.Button20Click(Sender: TObject);
```

```
begin
    x:=StrToFloat(Edit1.Text);
znak:=' s ';
// dla sinus jak dla innych działań
liczba:=False;
Edit1.Text:=' ';
    Edit1.SetFocus;
end;

procedure TForm1.Button17Click(Sender: TObject);
begin
    x:=StrToFloat(Edit1.Text);
// dla pierwiastka kwadratowego
znak:=' p ';
liczba:=False;
Edit1.Text:=' ';
    Edit1.SetFocus;
end;

procedure TForm1.Oprogramie1Click(Sender: TObject);
(* onClick dla "O programie" w MainMenu *)
begin
AboutBox.ShowModal;
// wyświetl formę drugą "O programie"
end;
```

```
procedure TForm1.Button21Click(Sender: TObject);
begin
  x:=StrToFloat(Edit1.Text);
  znak:=' c ';
  // dla cosinus jak wyżej
  liczba:=False;
  Edit1.Text:=' ';
  Edit1.SetFocus;
end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
  Case Key of
    '0'..'9', '#8', ',', '.', '- ':
      // wprowadzono cyfry, przecinek, BackSpace(kod #8 )
      ;
      // tylko średnik; nie rób nic,
      else
      // w przeciwnym razie
      Key:=#0;
      // zamień inne klawisze na znak pusty
  end;
end;

(*
- onKeyDown dla Edita – kiedy naciskamy klawisz w jego obrębie.
```

Procedura służy do wyeliminowania wprowadzania znaków innych niż cyfry od 0...9, przecinka i klawisza BackSpace (do wykasowania złego wpisu. Jego kod ASCII to #8).

- *jeśli wprowadzono 0...9, przecinek lub użyto klawisza BackSpace to*
- *nie rób nic. Wpisujemy sam średnik*
- *w przeciwnym razie (np. wpisuje litery)*
- *zamień inne klawisze na znak pusty (#0) *)*

```
procedure TForm1.fokus;  
begin  
  Edit1.SetFocus;  
  if ((not (liczba)) and (znak=' = ')) then  
    Edit1.Text:=' '  
  liczba:=True;  
end;
```

- (*
- *ustaw kursor w Edit1*
 - *jeśli ostatnio naciśnięto Button z cyfrą , a nie z działaniem i jednocześnie znak wykryty to „=”*
 - *wyczyść pole Edit1*
 - *ustaw zmienną „liczba” na True *)*

```
procedure TForm1.Button22Click(Sender: TObject);  
begin
```

```
x:=StrToFloat(Edit1.Text);  
znak:=' t ';  
// tangens  
liczba:=False;  
Edit1.Text:=' ';  
Edit1.SetFocus;  
end;  
end.
```

Program „Zmień tapetę, otwórz inny program”

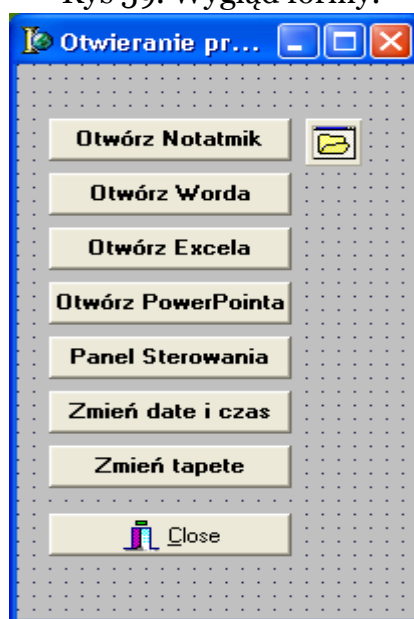
Program pokazuje jak z poziomu **Delphi** otwierać inne programy, zmieniać tapetę pulpitu, ustawiać czas systemowy.

Umieść na formie:

- **OpenDialog**
- siedem **Butonów**. Zmień ich **Caption** jak na rysunku.
- jeden **BitBtn**. Zmień jego właściwość **Kind** na **bkClose**.

We wszystkich procedurach używamy funkcji **ShellExecute**, więc do listy **Uses** należy dodać moduł **ShellAPI**.

Rys 59. Wygląd formy.



Kod źródłowy programu „Zmień tapetę, otwórz inny program”

```
unit Unit1;
```

```
interface
```

```
uses
```

```
    Windows, Messages, SysUtils, Variants, Classes, Graphics,  
    Controls, Forms,
```

```
    Dialogs, StdCtrls, ShellApi, Buttons;
```

```
type
```

```
TForm1 = class(TForm)
```

```
    Button1: TButton;
```

```
    Button2: TButton;
```

```
    Button3: TButton;
```

```
    Button4: TButton;
```

```
    Button5: TButton;
```

```
    Button6: TButton;
```

```
    Button7: TButton;
```

```
    BitBtn1: TBitBtn;
```

```
    OpenDialog1: TOpenDialog;
```

```
    procedure Button1Click(Sender: TObject);
```

```
    procedure Button2Click(Sender: TObject);
```

```
    procedure Button3Click(Sender: TObject);
```

```
    procedure Button4Click(Sender: TObject);
```

```
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
// dla "Otwórz Notatnik"
begin
    ShellExecute(Handle, ' open ', ' notepad.exe ', nil, nil, SW_SHOW);
end;

procedure TForm1.Button2Click(Sender: TObject);
// dla "Otwórz Worda"
begin
    ShellExecute(Handle, ' open ', ' WinWord.exe ', nil, nil,
SW_SHOW);
```

end;

```
procedure TForm1.Button3Click(Sender: TObject);
```

```
// dla "Otwórz Excela"
```

```
begin
```

```
ShellExecute(Handle, ' open ', ' excel.exe ', nil, nil, SW_SHOW);
```

```
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
```

```
// dla "Otwórz PowerPoint"
```

```
begin
```

```
ShellExecute(Handle, ' open ', ' powerpnt.exe ', nil, nil,  
SW_SHOW);
```

```
end;
```

```
procedure TForm1.Button5Click(Sender: TObject);
```

```
// dla "Panel Sterowania"
```

```
begin
```

```
ShellExecute(Handle, ' open ', ' control.exe ', nil, nil, SW_SHOW);
```

```
end;
```

```
procedure TForm1.Button6Click(Sender: TObject);
```

```
// dla "Zmień czas i date"
```

```
begin
```

```
ShellExecute(Handle, ' open ', ' control.exe ', ' timedate.cpl ', nil,  
SW_SHOW);
```

```
end;
```

(*

- *we wszystkich procedurach wykorzystujemy funkcję ShellExecute*
- *Handle to uchwyt, open – otwórz, następnie co, czyli podajemy plik.exe*
- *SW_SHOW znaczy, czy się wyświetli okno z programem *)*

```
procedure TForm1.Button7Click(Sender: TObject);
```

```
// dla "Zmień tapetę"
```

```
Var Plik:String;
```

```
begin
```

```
if OpenFileDialog1.Execute Then
```

```
Plik:=OpenDialog1.FileName;
```

```
SystemParametersInfo(SPI_SETDESKWALLPAPER, 0, Pchar(Plik),  
SPIF_UPDATEINIFILE OR SPIF_SENDWININICHANGE);
```

```
end;
```

```
end.
```

(* - zmiana tapety wybranej w OpenFileDialog *)

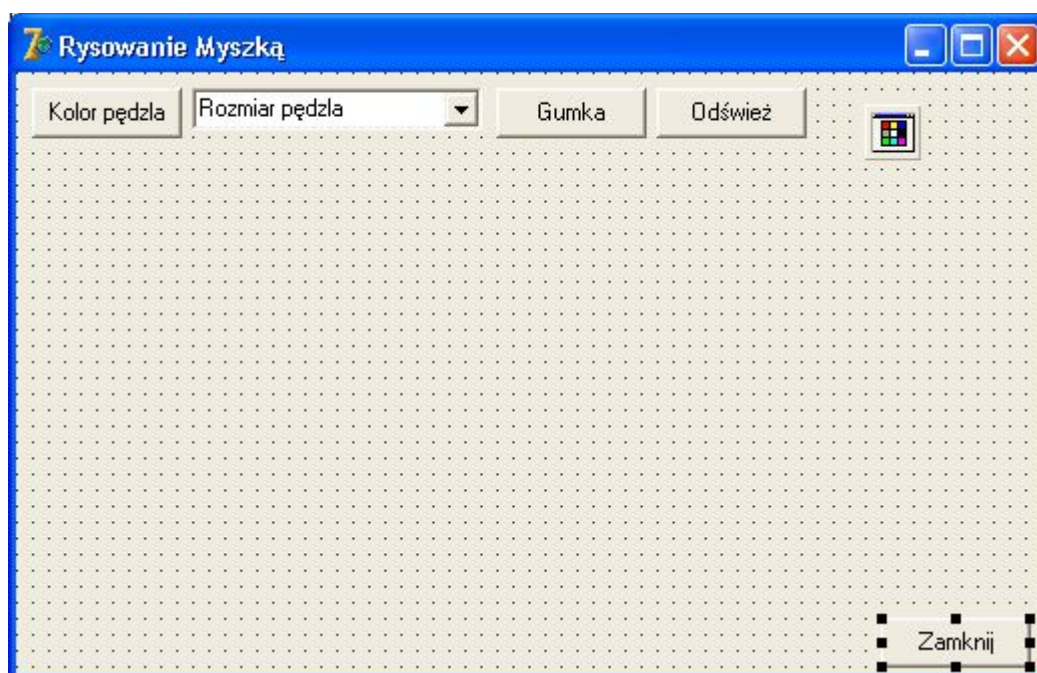
Program “Rysowanie myszką”

Program ten pokazuje, jak rysować myszką po płótnie formatki. Oczywiście można też zmienić rozmiar pędzla, jego kolor. Ma też opcje gumki.

Wstaw na firmę:

- **ComboBox**. Do **Items** wpisz kilka liczb będących rozmiarem pędzla. Do właściwości **Text** wpisz „Rozmiar pędzla”.
- cztery **Buttons**. Ich Caption na rysunku.
- **ColorDialog**. Okno wyboru koloru.

Rys 78. Wygląd formy „Rysowanie myszką”.



W programie wykorzystujemy zdarzenia ***onMouseDown***, ***onMouseMove*** i ***onMouseUp*** zachodzące odpowiednio podczas: wciśnięcia lewego przycisku myszy, przesuwania z wciśniętym lewym przyciskiem i podczas puszczenia lewego klawisza myszy, kiedy znajdujemy się nad danym komponentem. W tym przypadku będzie to formatka.

W sekcji ***Implementation*** zadeklarujemy zmienną typu Boolean, której zadaniem będzie wykrycie faktu naciśnięcia i puszczenia lewego przycisku myszki.

Opcja Gumki polega na rysowaniu kolorem domyślnym formy.

Kod źródłowy programu „Rysowanie myszką”

```
unit Unit1;

interface

uses

    Windows, Messages, SysUtils, Variants, Classes, Graphics,
    Controls, Forms,
    Dialogs, StdCtrls, ComCtrls, ToolWin, Buttons;

type

    TForm1 = class(TForm)
        Button1: TButton;
```

```
ColorDialog1: TColorDialog;
Button2: TButton;
ComboBox1: TComboBox;
Button3: TButton;
Button4: TButton;
procedure Button1Click(Sender: TObject);
    procedure FormMouseDown(Sender: TObject; Button:
TMouseButton;
    Shift: TShiftState; X, Y: Integer);
    procedure FormMouseMove(Sender: TObject; Shift: TShiftState;
X,
    Y: Integer);
    procedure FormMouseUp(Sender: TObject; Button:
TMouseButton;
    Shift: TShiftState; X, Y: Integer);
procedure Button2Click(Sender: TObject);
procedure ComboBox1Change(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
private
    { Private declarations }
public
    { Public declarations }
end;

var
    Form1: TForm1;
```

implementation

```
{$R *.dfm}
```

```
Var rysuj:Boolean;
```

```
// zmienna śledząca lewy przycisk
```

```
procedure TForm1.Button1Click(Sender: TObject);
```

```
begin
```

```
Close;
```

```
end;
```

```
procedure TForm1.FormMouseDown(Sender: TObject; Button:  
TMouseButton;
```

```
Shift: TShiftState; X, Y: Integer);
```

```
begin
```

```
Canvas.MoveTo(x,y);
```

```
rysuj:=True;
```

```
end;
```

```
(* W czasie wciskania lewego przycisku ustawiamy się w punkcie  
o współrzędnych X, Y , czy tam, gdzie klikniemy. *)
```

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift:  
TShiftState; X,
```

```
Y: Integer);
```

```
begin
```

```
if rysuj then
  Canvas.LineTo(x,y);
end;

(*
- jeśli zmienna rysuj ma wartość True to
- przesun się do punktu X, Y, czyli tam gdzie, przesuwasz
  wskaźnik myszy *)

procedure TForm1.FormMouseUp(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  rysuj:=False;
end;

(* Jak puścimy lewy przycisk, przypisz zmiennej rysuj wartość
False. *)

procedure TForm1.Button2Click(Sender: TObject);
begin
  // onClick dla "Kolor pędzla"
  if ColorDialog1.Execute Then
    Canvas.Pen.Color:=ColorDialog1.Color;
end;

(* Nadaj pędzlowi kolor wybrany w oknie ColorDialog1. *)

procedure TForm1.ComboBox1Change(Sender: TObject);
```

```
begin
```

```
Canvas.Pen.Width:=ComboBox1.ItemIndex;
```

```
end;
```

(Szerokość (Width) pędzla równa się liczbie z ComboBox1. *)*

```
procedure TForm1.Button3Click(Sender: TObject);
```

```
begin
```

```
// onClick dla "Odśwież"
```

```
Repaint;
```

```
end;
```

(odśwież, wyczyść *)*

```
procedure TForm1.Button4Click(Sender: TObject);
```

```
begin
```

```
// onClick dla "Gumka"
```

```
Canvas.Pen.Color:=clBtnFace;
```

```
end;
```

```
end.
```

(Do pędzla (tutaj: Gumki) przypisz kolor formy. *)*

Rys 79. Program „Rysowanie myszką” w działaniu.



Jak skorzystać z wiedzy zawartej w pełnej wersji ebooka?

Więcej praktycznych programów znajdziesz w pełnej wersji ebooka.

Zapoznaj się z opisem na stronie:

<http://delphi-programy.zlotemysli.pl>

Naucz się pełniej wykorzystywać możliwości Delphi!



**Poleć znajomemu e-booka
i zarób 50% jego wartości**



**Kupuj e-booki za punkty,
nie za złotówki**

POLECAMY TAKŻE PORADNIKI:

[Programuję w Delphi i C++ Builder](#) - Mirosław J. Kubiak



Jak szybko nauczyć się programowania w dwóch różnych językach?

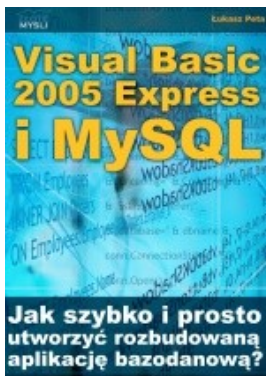
Standardową metodą jest uczenie się programowania "język po języku". A co powiesz na dwa w jednym? Co gdybyś przy okazji zagłębiania się w szczegóły danej instrukcji w Delphi dowiedział się od razu, jak to samo robi się w C++ Builderze?

Więcej o tym poradniku przeczytasz na stronie:
<http://podstawy-delphi-builder.zlotemysli.pl>

"(...) Ta pozycja zawiera kompendium programowania obiektowego w jednym tomie. Ba! W jednym pliku. Jest świetna dla uczących się (...) jak i dla kogoś, kto na bieżąco nie programuje, ale potrzebuje odświeżyć sobie informacje (...)"

Stanisław Janiak, Teleinformatyk, 50 lat.

[Visual Basic 2005 Express i MySQL](#) - Łukasz Peta



Jak szybko i prosto utworzyć rozbudowaną aplikację bazodanową?

Ebook "[Visual Basic 2005 Express i MySQL](#)" uczy zarówno składni języka Visual Basic jak również zasad tworzenia aplikacji opartych o bazę danych MySQL dla systemu Windows w tym języku, a został napisany głównie z myślą o początkujących programistach

Więcej o tym poradniku przeczytasz na stronie:
<http://visual-basic.zlotemysli.pl>

"Dosyc, ze e-book to jeszcze dodatkowo kody i przyklady aplikacji do nauki. Bardzo wartosciowy e-book. Czysto i prosto przekazana wiedza. Polecam."

David 27 lat, programista

**Zobacz pełen katalog naszych praktycznych poradników
na stronie www.zlotemysli.pl**