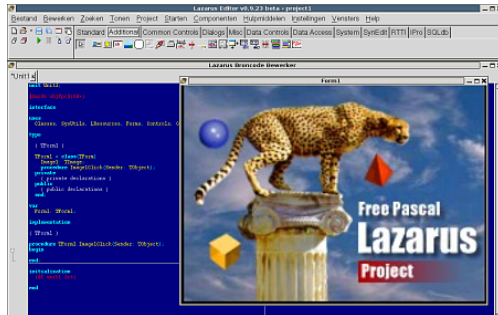


Lazarus

Programowanie w Delphi



Delphi

- Delphi jest to język oparty na Pascal.
- Twórcą jego jest firma Borland. Pierwotnie nosił nazwę Object Pascal (do 2006).
- Jest to język prosty i popularny.
- Wykorzystuje środowisko graficzne, co pozwala na napisanie prostych i efektownych programów.
- Posługuje się technologią RAD (*Rapid Application Development*), co ułatwia tworzenie interfejsu programu.

2

Linki do stron o Delphi

- <http://www.borland.pl/delphi/> - strona producenta
- <http://www.swissdelphicenter.ch/en/> - baza wiedzy na temat Delphi
- <http://www.torrey.net/> - olbrzymia baza komponentów oraz przykładowych kodów źródłowych
- <http://delphi.icm.edu.pl/> - komponenty dla Kyliksa oraz wszystkich wersji Delphi i C++ Buildera
- <http://www.delphi.org.pl> - Polska Grupa Użytkowników Delphi
- <http://delphi.about.com/> - mnóstwo kodów źródłowych
- <http://www.unit1.pl/> - Strona poświęcona programowaniu w Delphi
- <http://4programmers.net/Delphi> - Strona poświęcona programowaniu w Delphi
- <http://4programmers.net/Delphi/Kompendium> - książka "Delphi 7. Kompendium programisty"

3

Kompilatory Delphi

Istnieją trzy kompilatory języka Delphi:

1. Borland Delphi

- Edytor twórcy Delphi. Pierwsza wersja wyszła w 1995 do Windows 3.1. Obecnie istnieje 10 wersja tego środowiska.

2. Kylix

- Kompilator do Linuksa. Wyszły trzy wersje i projekt zarzucono.

3. Lazarus

- Darmowy projekt Open Source.

4

Lazarus

- *Lazarus* to zintegrowane środowisko programistyczne oparte o kompilator *Free Pascal*.
 - Jest to wzorowane na Delphi wizualne środowisko programistyczne oraz biblioteka *Lazarus Component Library* (LCL), która jest odpowiednikiem VCL.
- *Lazarus* jest zgodny z Delphi. Pozwala na rozwijanie programów, podobnie jak w Delphi, na wszystkich platformach obsługiwanych przez FPC.
 - *Lazarus* i *Free Pascal* starają się, aby raz napisana aplikacja kompilowała się wszędzie. Ponieważ dostępny jest dokładnie taki sam kompilator, nie trzeba wprowadzać żadnych zmian, aby otrzymać taki sam produkt dla różnych platform.
 - Program napisany w środowisku *Lazarus* można bez żadnych zmian skompilować dla dowolnego obsługiwanego procesora, systemu operacyjnego i interfejsu okienek.
- Program jest udostępniany na licencji GNU GPL, natomiast biblioteki na zmodyfikowanej licencji LGPL (umożliwia to wykorzystanie *Lazarusa* w projektach o zamkniętym kodzie).

5

Lazarus – cechy

- Lazarus posiada:
 - kompilator języka Pascal (FreePascal)
 - edytor kodu źródłowego
 - RAD (Rapid Application Development) - szybkie tworzenie programu
 - wizualne tworzenie form (okien programu)
 - RTL - zestaw gotowych komponentów
 - możliwość generowania kodu dla
 - kilku platform operacyjnych (win32, linux, mac)
 - i sprzętowych (pentium, PowerPC, Mac)

6

Lazarus – różne platformy

- Częścią Lazarusa jest Interfejs „*Widgets*”, który umożliwia płynne przejście na inne platformy z zastosowaniem różnych interfejsów okienek.
- Aktualnie w różnych stadiach zaawansowania są interfejsy do następujących platform:
 - Windows
 - Unix
 - Mac OS X
 - Środowiska przenośne:
 - Windows CE
 - Qtopia for Linux-based PDAs
 - PalmOS (w przyszłości)
 - Symbian OS (w przeszłości)

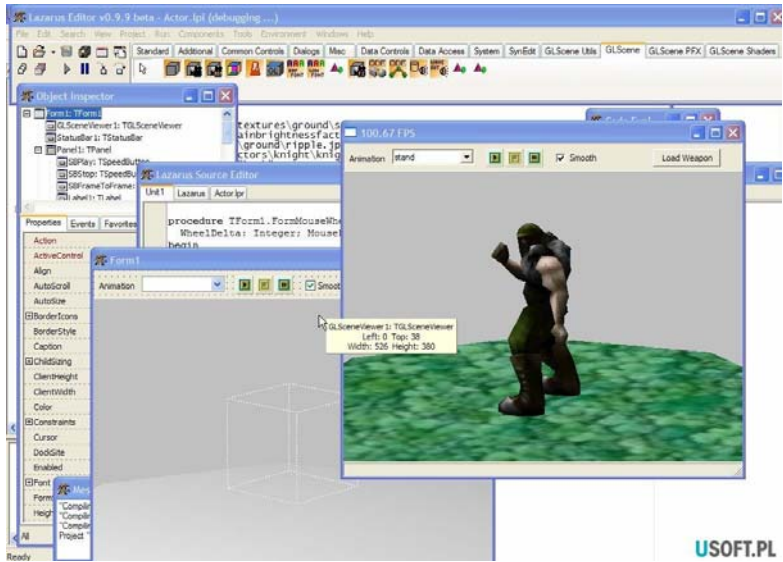
7

Lazarus – materiały WWW

- <http://pl.wikipedia.org/wiki/Lazarus> - Artykuł w Wikipedii
- <http://www.lazarus.freepascal.org/> - strona projektu Lazarus
- <http://lazarus-ccr.sourceforge.net/> - dokumentacja Lazarusa
- http://wiki.lazarus.freepascal.org/index.php/Lazarus_Documentation - Wiki dokumentacja do Lazarusa
- <http://www.skinhat.com/lazarus/> - Biblioteka GL do Lazarusa
- <http://www.freepascal.org/> - Free Pascal

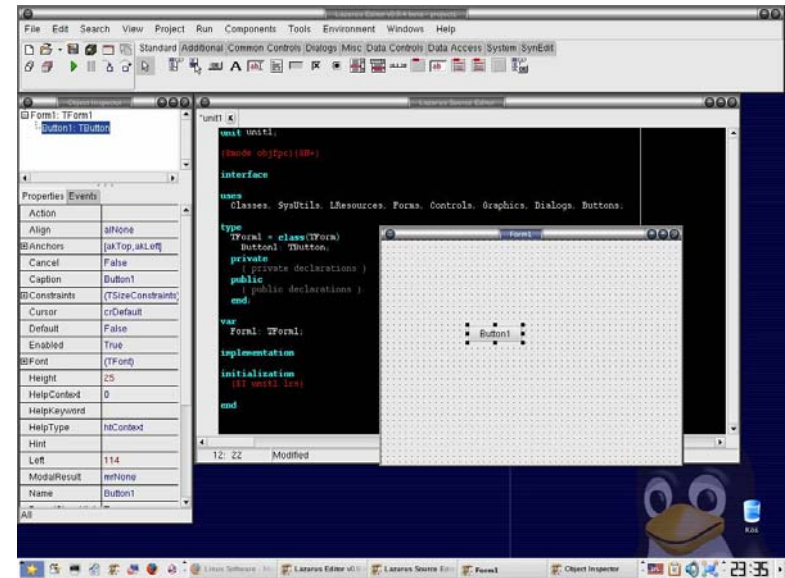
8

Lazarus na Windows



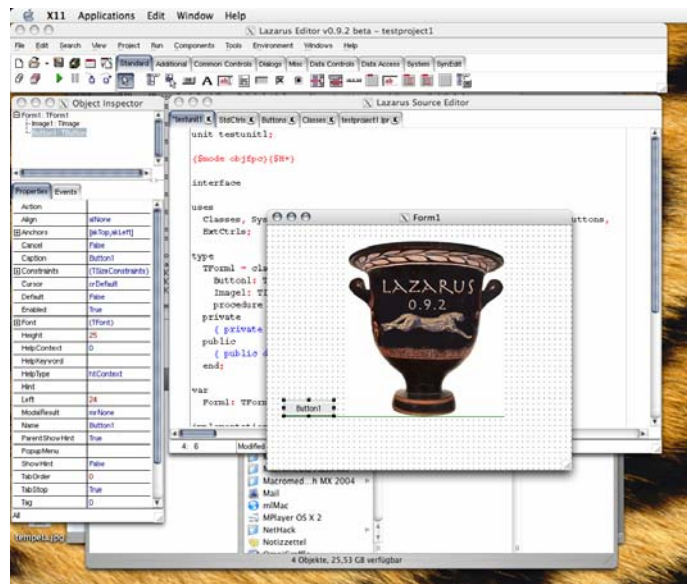
9

Lazarus na Ubuntu



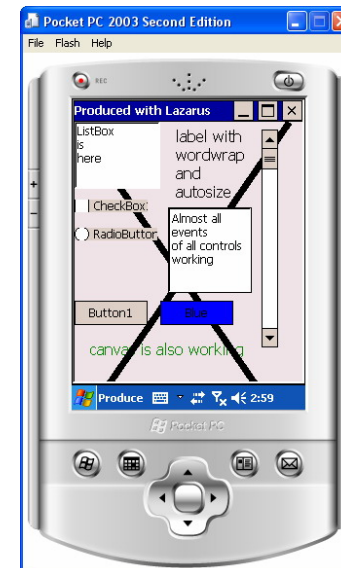
10

Lazarus na Mac OS X

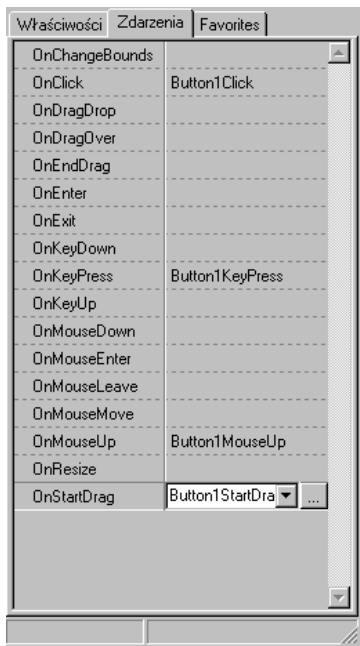


11

Lazarus na Pocket PC



12



Zdarzenia

- Zdarzenia to podprogramy, które reagują na określone wydarzenia związane z danym komponentem.
- Zakładka ta składa się z dwóch kolumn:
 - Lewa zawiera nazwy poszczególnych zdarzeń
 - Prawa zawiera procedury i funkcje przypisane do nich.
- Zdarzenie **OnClick** odpowiada sytuacji, gdy dany komponent zostanie kliknięty myszką. Przypisany podprogram wykona daną operację.
- Różnym zdarzeniom można przypisać ten sam podprogram.

17

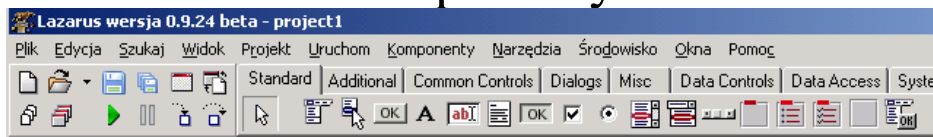


Favorites

- Trzecia zakładka zawiera ulubione właściwości i zdarzenia, które są często używane w odniesieniu do danego komponentu.

18

Komponenty



- Środowisko Delphi zawiera zbiór podstawowych komponentów wykorzystywanych w programach.
- Są to np. przyciski, okienka, napisy, suwaki, pola wyboru, menu, tabelki itp.
- Pogrupowane są w kilku zakładkach, co pozwala je szybko wyszukać.
- Istnieje możliwość dodania nowych komponentów i stworzenia własnych.

19

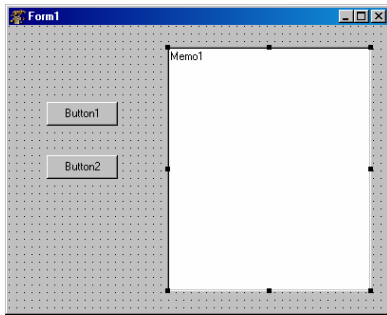
Komunikaty



- Komunikaty to opisy, uwagi, ostrzeżenia i błędy wygenerowane przez kompilator Lazarus.
- Jest to podstawowe źródło informacji w razie problemów z uruchomieniem programu.

20

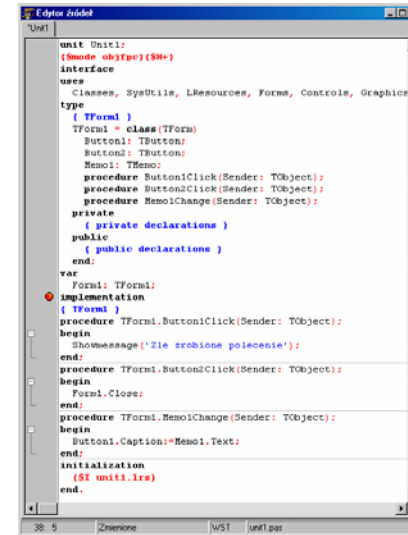
Tworzony program



- Okienko to przedstawia program który jest tworzony.
- Jest to tryb WYOSIWYG (*What You See Is What You Get*).
- Pozwala to na prostą i efektywną pracę.
- Zmianę właściwości i dodanie kodu realizujemy klikając na dany komponent.

21

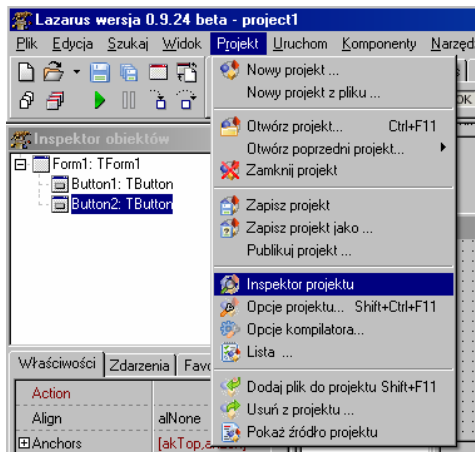
Edytor źródeł - Kod programu



- To okno zawiera kod tworzego programu.
- Pozwala na ręczną edycję kodu.
- Poszczególne części są wyróżnione innym kolorem lub tłustym drukiem.
- Domyślnie są uwzględnione wcięcia w kodzie.

22

Menu Edytora



- Klasyczne menu edytora.
- Pod nim zebrano kilka najczęściej używanych poleceń jak:
 - zapisz,
 - uruchom,
 - nowy projekt.

23

Program w języku Delphi

24

Tworzenie programów w Lazarus

- Program w języku Delphi składa się z kilku plików źródłowych:
 - *.pas – plik z kodem źródłowym programu
 - *.lpr – Plik *Lazarus Project*. Zawiera budowę całego programu.
 - *.lpi – Plik *Lazarus Project Information*. Zawiera dokładne dane o projekcie.
 - *.lfm — plik Lazarus Form. Zawiera opis formularza.
 - *.lrs — plik zasobów.
- Z reguły nazwa:
 - projektu zaczyna się od słowa `project`
 - pliku z kodem od słowa `unit`

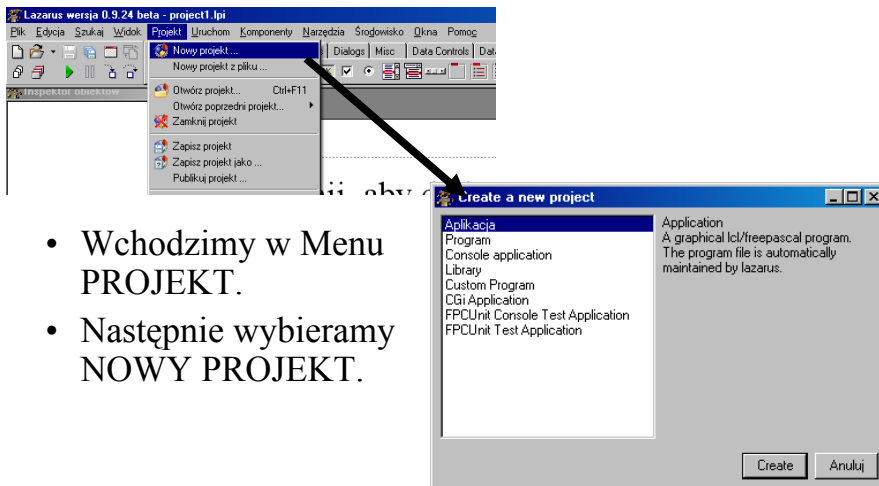
25

Zapisywanie projektów

- Każdy program składa się z kilku plików, które mają podobne nazwy.
 - Może to wprowadzić zamieszanie lub projekty mogą się skrzyżować.
- Należy więc utworzyć oddzielne katalogi dla każdego projektu.
 - Tworzymy je w `C:\lazarus\projectsessions\`
 - Można je też zapisywać na pendrive i nosić ze sobą.
- Powinny składać się z numeru projektu i imienia (lub nazwiska) twórcy.
 - Mają one mieć wzór typu: `projekt2Jacek`.

26

Nowy projekt

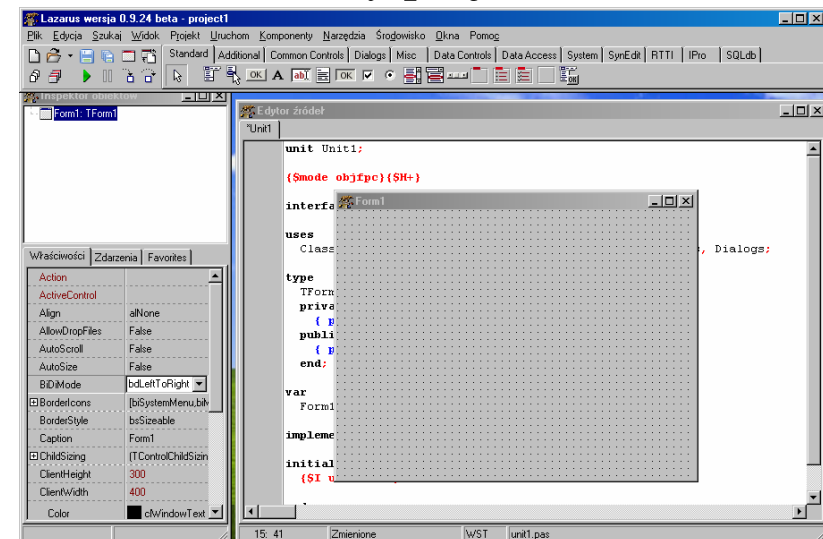


- Wchodzimy w Menu **PROJEKT**.
- Następnie wybieramy **NOWY PROJEKT**.

- Z następnego okna wybieramy pierwszą pozycję – **APLIKACJA**.

27

Nowy projekt cz.2



28

Budowa programu

29

Budowa programu cz.1

Program składa się z kilku wyróżnionych i wymaganych części

- **unit** - Unit to część projektu zapisywana w oddzielnym pliku.
- **interface** - Metody z których korzysta Unit. Są to różne właściwości i operacje jakie może wykonywać.
- **uses** - składniki używane przez dany Unit
- **type** – definicja występujących typów. Deklaracja komponentów, procedur i funkcji przez nie używanych.
 - **private** - zmienne prywatne niedostępne z innych unitów
 - **public** - zmienne dostępne z innych unitów
- **var** - zmienne globalne występujące w Unicie
- **implementation** - implementacja metod– procedury i funkcje
 - **procedure** - procedura
 - **function** - funkcja
- **initialization** – polecenie inicjalizacji zasobów edytora

30

Budowa programu cz.2

```
unit Unit1; //początek programu – nazwa Unita
{$mode objfpc}{$H+} //polecenie dla kompilatora
interface
uses //lista zasobów używanych przez Unit. Część jest standardowa i dodawana automatycznie.
Classes, SysUtils, LResources, Forms, Controls, Graphics, Dialogs, StdCtrls;
type
{ TForm1 } //komentarz – definicja typu TForm1
TForm1 = class(TForm) //obiekt TForm1 należy do klasy TForm
Button1: TButton; //komponent Przycisk
Button2: TButton;
Edit1: TEdit; //komponent Okienko edycyjne
Edit2: TEdit;
Label1: TLabel; //komponent Etykieta
procedure Button1Click(Sender: TObject); //deklaracja procedura
procedure Button2Click(Sender: TObject);

private //deklaracja zmiennych prywatnych– niedostępnych z innych Unitów
{ private declarations }
public //deklaracja zmiennych publicznych – dostępnych z innych Unitów
{ public declarations }
end;

var //zmienne
Form1: TForm1;
x,y,z:real;
```

31

Budowa programu cz.3

```
Implementation //Część implementacyjna Unitu
{ TForm1 } //komentarz – implementacja metod TForm1
procedure TForm1.Button1Click(Sender: TObject); //definicja procedury
begin
x:=StrToFloat(Edit1.Text);
y:=StrToFloat(Edit2.Text);
z:=x+y;
Edit3.Text:= FloatToStr(z);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
x:=StrToFloat(Edit1.Text);
y:=StrToFloat(Edit2.Text);
z:=x-y;
Edit3.Text:= FloatToStr(z);
end;

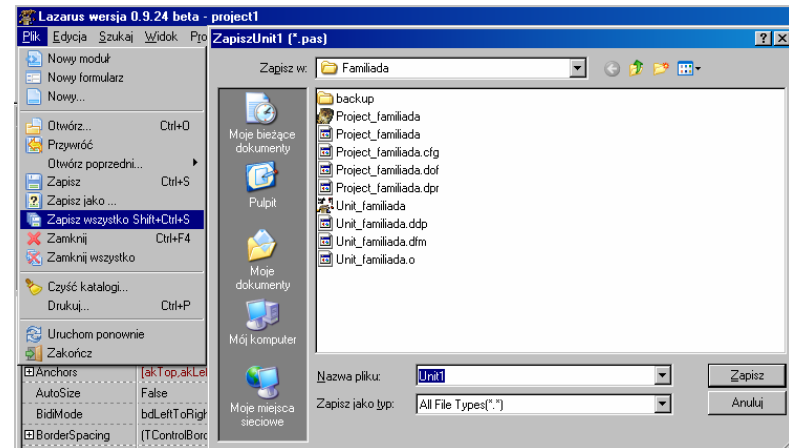
initialization //polecenie dla kompilatora – Inicjalizacja zasobów edytora
{$I unit1.lrs}
end. //koniec Unitu
```

32

Zapisywanie, kompilacja i uruchamianie

33

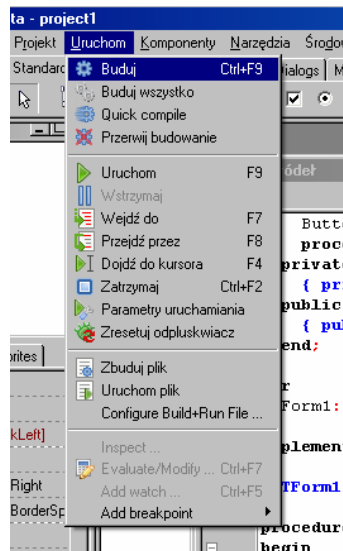
Zapisanie programu



- Opcje zapisu:
 - Zapisz (CTRL+S)
 - Zapisz wszystko (CTRL+Shift+S)

34

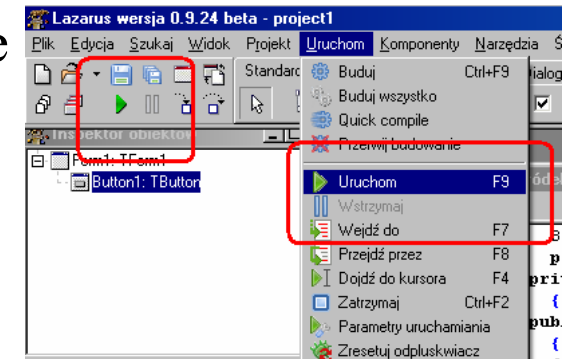
Kompilacja



- Wchodzimy w Menu URUCHOM (**RUN**)
 - Wybieramy opcję BUDUJ (**BUILD**)
- Kompilację można wywołać klawiszami CTRL + F9.

35

Uruchomienie programu



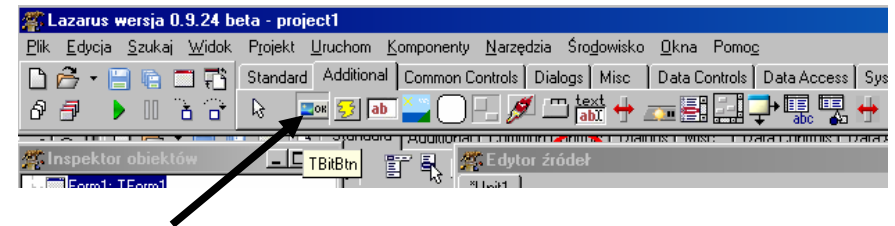
- Wchodzimy w Menu URUCHOM (**RUN**)
 - Wybieramy opcję URUCHOM (**RUN**)
- Lub naciskamy zieloną strzałkę na pasku zadań.
- Uruchomienie można wywołać klawiszem F9.
- Chcąc zakończyć działanie należy wybrać pole WSTRZYMAJ lub nacisnąć znak pauzy.

36

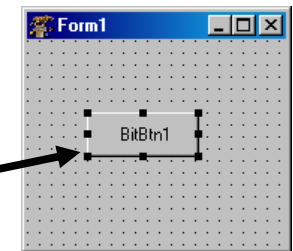
Wstawianie komponentów i edycja ich właściwości

37

Wybór i wstawienie elementu

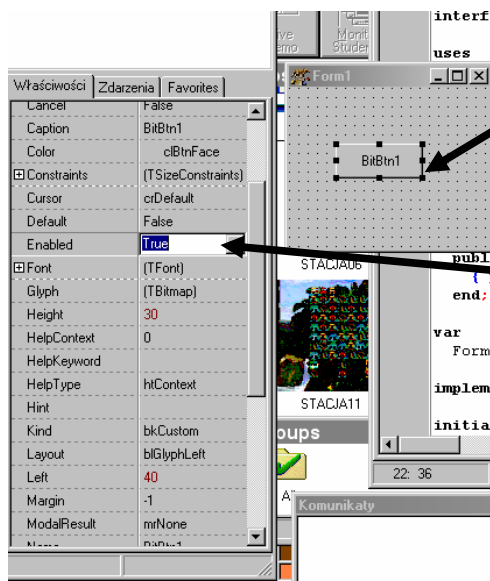


- Należy wybrać komponent na odpowiedniej zakładce i na niego kliknąć.
- Następnie wstawić go w odpowiednim miejscu na formatce.



38

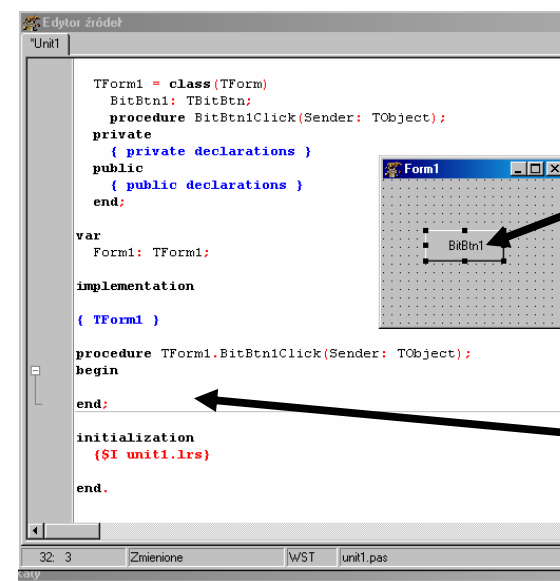
Edycja właściwości



- Należy kliknąć na komponent, by móc edytować jego właściwości lub zdarzenia.

39

Edycja kodu programu



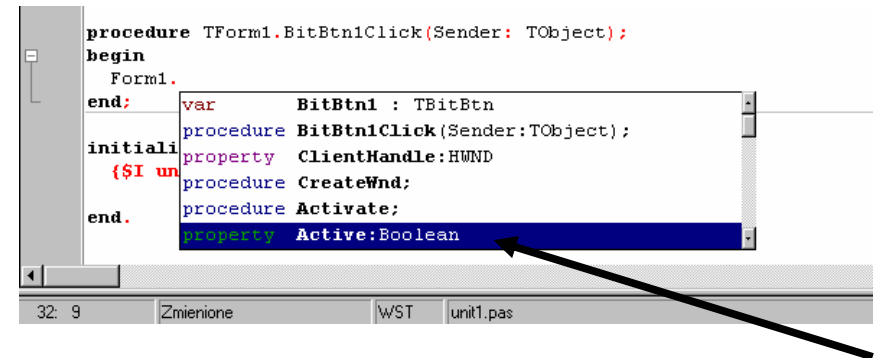
- Należy kliknąć na komponent, by móc edytować jego zdarzenia lub dodać polecenia wykonywane po kliknięciu na komponent.

40

Edycja właściwości obiektów w kodzie programu

41

Automatyczna podpowiedź

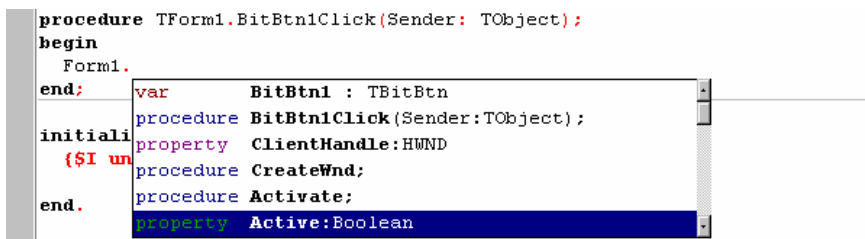


```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Form1.
end;
var    BitBtn1 : TBitBtn
procedure BitBtn1Click (Sender:TObject);
property ClientHandle:HWND
procedure CreateWnd;
procedure Activate;
property Active:Boolean
```

- W trakcie pisania wystarczy wpisać nazwę obiektu i kropkę.
- Pojawi się wtedy podpowiedź zawierająca zbiór właściwości, funkcji i procedur, które może zrealizować ten obiekt.
- Wystarczy więc poszukać i wybrać odpowiedni.

42

Wstawienie odpowiedniej wartości



```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
  Form1.
end;
var    BitBtn1 : TBitBtn
procedure BitBtn1Click (Sender:TObject);
property ClientHandle:HWND
procedure CreateWnd;
procedure Activate;
property Active:Boolean
```

- Podpowiedź pokazuje jednocześnie typ zmiennych, które może przybrać dana właściwość.
- W powyższym przykładzie są to wartości logiczne (*true* lub *false*).

```
procedure TForm1.BitBtn1Click (Sender:
  TObject);
begin
  Form1.Visible:=false;
end;
```

43

Metody danego komponentu

- **var** – komponenty znajdujące na aktualnym komponentcie.
- **property** – właściwość komponentu
- **procedure** – procedura wykonywana przez ten komponent
- **function** - funkcja wykonywana przez ten komponent

44