**Mówiąc o Delphi mówimy jednocześnie o środowisku programowania jak i języku. Mianowicie:**

**Delphi** jest to język programowania, którego można używać w środowiskach firmy Borland, Embarcadero, Microsoft (Delphi Prism), oraz w środowisku Lazarus. Narzędzia te są zintegrowanymi środowiskami programistycznymi,w skrócie ( IDE) czyli aplikacją lub zespółem aplikacji (środowisko) służących do tworzenia, modyfikowania, testowania i konserwacji oprogramowania.). Narzędzia są typu RAD(**Rapid Application Development** ) co oznacza "szybkie tworzenie aplikacji". RAD to ideologia i technologia polegająca na udostępnieniu programiście dużych możliwości prototypowania oraz dużego zestawu gotowych komponentów (np. zapewniających dostęp do bazy danych). Umożliwia to uzyskanie pewnego efektu już w pierwszych krokach programistycznych, jednocześnie stanowi poważne zagrożenie dla projektów o większych rozmiarach ze względu na łatwość nieprzemyślanego modyfikowania.). Działaja zgodnie z zasadą dwustronnej edycji(**Dwustronna edycja to** funkcja środowiska programistycznego odzwierciedlająca zmiany dokonywane na zasobach w kodzie programu i czasem odwrotnie).

Delphi jest jednym z bardzo popularnych środowisk programistycznych. Posiada ogromne możliwości w zakresie tworzenia aplikacji

Dawniej język Delphi był nazywany Object Pascalem, lecz w świadomości społecznej programistów na całym świecie Delphi zaczęło być postrzegane z czasem jako osobny język programowania i dlatego też przy okazji premiery Delphi 6 w roku 2002 w oficjalnej dokumentacji programu została użyta po raz pierwszy nazwa "Delphi language". Standard języka Delphi obejmuje wiele bogatych funkcjonalnie klas, których nie ma w standardzie Object Pascala, a ponadto umożliwia programowanie wizualne z wykorzystaniem predefiniowanych komponentów.

Kompilatory języka Delphi kompilują kod Delphi do natywnego (właściwy danemu środowisku informatycznemu) kodu x86 lub zarządzanego kodu .NET.

Programowanie w Delphi oparte jest na komponentach wizualnych (VCL, (Visual Component Library)), Tworzenie programu odbywa się na zasadzie programowania obiektowego. Biblioteka komponentów oferuje obiekty, które można umieszczać w obszarze projektowania. Dobierając komponenty do naszych potrzeb, dopasowując ich własności oraz pisząc funkcje obsługi zdarzeń tworzymy program.

Środowisko Borland Delphi pozwala w łatwy i przystępny sposób tworzyc aplikacje multimedialne, graficzne, dźwiękowe, zawierające animacje, korzystające z różnego rodzaju baz danych, zawierające mechanizmy OLE, kontrolki OCX, pozwalające na prace w sieci lokalnej, internecie i wiele innych. Można dotrzeć do urządzeń peryferyjnych, portów LPT, RS232, USB, portów wejścia \ wyjścia, do dowolnych plików a nawet dowolnych punktów ekranu.

Posiada dziedziczenie po jednej klasie i po wielu interfejsach, co sprawia, że programy pisane w Delphi są czytelniejsze niż programy pisane w językach pozwalających na wielodziedziczenie.

Środowisko Delphi zawiera także zaawansowane funkcje matematyczne, logiczne, graficzne oraz systemowe które pozwalają tworzyć zaawansowane aplikacje oraz narzędzia znajdujące zastosowanie praktycznie we wszystkich gałęziach oprogramowania komputerowego.

Bazową klasą obiektu jest zawsze metaklasa TObject. Obiekty są przekazywane przez referencje (tak jak w Javie i C#)

Delphi generuje kod źródłowy, po skompilowaniu którego tworzony jest wykonywalny program dla systemu Windows. Rozszerzenie.

ROZWÓJ JĘZYKA

Wraz z rozwojem technologii informatycznych język Delphi ewoluował, zachowując jednak kompatybilność wstecz. Każdy projekt napisany w poprzedniej wersji języka jest kompilowalny w najnowszych kompilatorach.

Znaczna ewolucja w edycji w Delphi nastąpiła w roku 2009. Zmiany, które zaszły, to:

- Pełne wsparcie języka dla Unicode (w tym dla VCL)
- Wprowadzenie klasy TStringBuilder dla szybszej konkatenacji łańcuchów, połączenie dwóch ciągów znaków
- Wprowadzenie typów generycznych, możliwe do użycia w szerokim polu zastosowań, akceptujące różne typy argumentów, przetwarzanych danych itp.;
- Klasa TObject posiada wirtualne metody ToString, GetHashCode, i Equals
- Metody anonimowe
- Pełne wsparcie dla formatu plików PNG dla klasy TImage

W wersji 2010 dodano do języka możliwość używania atrybutów, oraz wprowadzono kilka nowych unitów do bibliotek standardowych języka.

**Historia środowiska programistycznego Delphi**

Delphi, opracowane przez firmę Borland, jest następcą środowiska Turbo Pascal, będącego połączeniem kompilatora języka Pascal ze środowiskiem edycyjnym.

- Pierwsza wersja (16-bitowa) Delphi ukazała się w 1995 roku i przeznaczona była do pracy w Microsoft Windows 3.1.
- Kolejna wersja – Delphi 2 – ukazała się w roku 1996 i była środowiskiem 32-bitowym, generującym kod dla Microsoft Windows 95.
- Delphi 3 ukazało się w roku 1997 i wprowadziło do środowiska następujące elementy: pakiety, rozproszoną obsługę baz danych, wspomaganie tworzenia aplikacji internetowych, wspomaganie tworzenia kontrolek ActiveX.
- Delphi 4 dodało wiele nowych usprawnień IDE (jak na przykład możliwość dokowania okien), oraz usprawnienia w samym języku programowania (na przykład przeciążanie funkcji oraz metod).
- W wersji 6 pojawiła się m.in. możliwość tworzenia przeźroczystych okien (tylko Windows 2k/XP).
- Wersja 7 wniosła nowy komponent pozwalający na używanie stylów wizualnych z Windows XP w tworzonych programach. W roku 2001 powstała edycja przeznaczona dla Linuksa nazwana Kylix.
- Wersja 8 (rok 2003) wprowadziła Delphi w środowisko .NET, po raz pierwszy od powstania uległo też zmianie IDE, upodobniając się do Microsoft Visual Studio .NET.
- Delphi Prism – jest to wtyczka do IDE Microsoft Visual Studio, wraz z kompilatorem – stanowiący nowe środowisko programistyczne. Pozwala na pisanie aplikacji .NET dla Windows, Linux i Mac. Nowy kompilator jest oczekiwaną alternatywą dla języka C# – obsługuje .NET 1.1, 2.0, 3.0, 3.5, WinForms, WPF, Silverlight, ASP.NET i LINQ. Nie obsługuje Compact Framework. W języku Delphi Prism nie można używać już VCL.NET. \
- Delphi 2009 – Delphi dla Windows obsługujące Win32. Do języka wprowadzono typy generyczne, metody anonimowe i obsługę składniową dla Unicode ze strony języka.

- Delphi 2010 - wydano 25 sierpnia 2009 roku. Jest to druga wersja środowiska Delphi, która obsługuje Unicode. Nie niesie dużych zmian w stosunku do Delphi 2009. Zawiera nowy, ulepszony system , który skutkuje tworzeniem większych plików wykonywalnych niż w poprzednich wersjach Delphi. Delphi 2010 posiada wsparcie dla Windows 7, oraz dla ekranów dotykowych i gestów (nowe kontrolki VCL)
- 
- Delphi XE - wydano 30 sierpnia 2010 roku. Dodano kilka narzędzi do środowiska takich jak AQtime (wersja standard), CodeSite (wersja express), Beyond Compare, zapewniono integrację z Subversion w IDE, dodano kontrolki do współpracy z Windows Azure oraz do przetwarzania w chmurze obliczeniowej Amazon EC2.
- 
- Delphi XE Starter Edition - edycja Delphi XE udostępniona 27 stycznia 2011, skierowana dla hobbystów, studentów i małych firm - dostępna za niższą cenę (około 190 $)

CECHY

- Tworzenie 32-bitowego kodu programu.
- Praca w trybie wizualnego projektowania.
- Oparty na języku Object Pascal, lecz bardziej rozbudowany.
- Prosty i przejrzysty kod źródłowy.
- Bardzo dobry system pomocy.
- Duża liczba dodatkowych komponentów.
- Nowoczesność oraz wsparcie dla najnowszych technologIi.

.
wspomaganie dla obsługi relacyjnych systemów bazodanowych
dwustronna edycja
szybki, efektywny kompilator

# Przykłady oprogramowania napisanego w całości bądź częściowo przy użyciu języka Delphi

**Ad-Aware** – program firmy Lavasoft służący do wyszukiwania programów szpiegujących.

**Dev-C++** – zintegrowane środowisko programistyczne, obsługujące języki C i C++, na licencji GPL, dla systemów rodziny Windows i Linux.

**Skype** – komunikator internetowy, oparty na technologii peer-to-peer.

**Total Commander** – menedżer plików działający w środowisku Microsoft Windows,

**Odkurzacz** − aplikacja umożliwiająca pozbycie się zbędnych plików na dysku w systemie Windows.

**BUDOWA PROGRAMU**

**1** - Forma (Form) Jest to okienko naszej przyszłej aplikacji. Umieszczamy w nim komponenty. Po skompilowaniu i uruchomieniu programu zobaczymy dokładnie to co zaprojektowaliśmy.
**2** - TreeView służy do przeglądania zawartości naszego programu. Pokazuje strukturę umieszczonych w nim komponentów. Jest bardzo przydatny przy rozbudowanych aplikacjach
**3** - Object Inspector Będziemy z niego bardzo często korzystać. Zawiera właściwości (Properties) oraz zdarzenia (Events) przypisane do każdego komponentu. Pozwala na ich modyfikację.
**4** - Biblioteka komponentów (VCL) Zawiera wszystkie komponenty możliwe do użycia. Komponenty ułożone są pod względem spełnianych funkcji. Aby zastosować komponent w programie należy przeciągnąć jego ikonę na formę.
**5** - Menu Zawiera standardowe opcje takie jak zapis lub odczyt oraz wiele innych.

Elementem nie widocznym na ilustracji jest *edytor kodu*. To właśnie w jego oknie wpisujemy kod programu. Aby wywołać okno edytora kodu naciskamy klawisz F12.

**Szkielet programu**
Delphi po uruchomieniu tworzy automatycznie szkielet nowej aplikacji. Jak zapewne zauważyłeś w poprzedniej lekcji duża ilość kodu programu powstaje automatycznie podczas projektowania aplikacji, bez potrzeby ingerencji ze strony użytkownika.
Każda aplikacja zawiera blok główny stanowiący podstawę programu. Aby go zobaczyć należy wybrać z menu Project>View Source . Naszym oczom ukarze się kod którego nie należy modyfikować. Jest on automatycznie tworzony przez Delphi i niech tak pozostanie.
W środowisku wizualnym dużo większe znaczenie od bloku głównego mają moduły (Units). Każde okienko naszej aplikacji posiada swój moduł w którym zapisane są wszystkie procedury i funkcje dotyczące tego okna oraz komponentów na nim umieszczonych. Standardowo struktura modułu wygląda następująco:

```
unit nazwa_modułu;

interface

uses ...
const ...
type ...
procedure ...
function ...

private { Private declarations }

public { Public declarations }

end;

var ...

implementation

...

end. //Koniec modułu
```

Każdy moduł rozpoczyna się od nagłówka *Unit* zawierającego nazwę modułu. Następnie następuje sekcja *Interface*. Jest to część opisowa zawierająca następujące cześci:
*Uses* - zawiera informacje z jakich bibliotek lub innych modułów korzystamy
*const* - zawiera deklaracje stałych
*type* - tutaj znajdują się deklaracje typów (np. TForm1 = class(TForm) ), umieszczamy tu też nagłówki procedur i funkcji przeznaczonych do udostępnienia (np. procedure Button1Click(Sender: TObject); )
*Private i Public* - sekcje te informują kompilator że deklaracje w nich zawarte mają być tylko do użytku lokalnego (private) lub globalnego (public)
*var* - tutaj deklarujemy zmienne
*implementation* - w tej sekcji zgromadzone są procedury i funkcje naszego programu
*end.* - "end z kropką" zawsze oznacza koniec modułu lub rogramu.

**Co to jest VCL**
VCL to  (Biblioteka Wizualnych Komponentów). Jest jednym z najważniejszych elementów Delphi. To właśnie niej zawdzięczamy łatwość i szybkość projektowania aplikacji. Biblioteka VCL składa się z gotowych napisanych wcześniej komponentów które podczas kompilacji są dołączane do wykonywalnego pliku naszego programu. Po skompilowaniu program nie wymaga żadnych dodatkowych plików. Może być uruchamiany na dowolnym komputerze bez względu na rodzaj zastosowanych komponentów. Standardowe komponenty VCL nadają się do większości zastosowań, gdybyśmy jednak potrzebowali jakiś specjalnych funkcji możemy doinstalować dodatkowe komponenty (w sieci jest ich bardzo wiele) lub napisać własny.

**Metody, właściwości, zdarzenia**
Komponent jest obiektem przeznaczonym do wykonywania określonych fonkcji. Różne komponenty posiadają rózne funkcje i tym samym przeznaczenie. Zbiór komponentów to coś w rodzaju klocków, dobierając odpowiednie możemy w krótkim czasie zbudować aplikację.

Komponenty posiadają określone metody, właściwości oraz zdarzenia. Pojęcia te zostały wprowadzone aby zdefiniować sposób komunikowania się komponentu z otoczeniem. Umieśćmy teraz na formie komponent Button z palety Standart. Pojawi się on na formie w postaci prostokatnego przycisku z napisem Button1. Zaznacz go pojedyńczym kliknięciem. W okienku Object Inspector pojawią się jego właściwości (Properties) a na drugiej karcie zdarzernia (Events).

Opis zdarzeń:
a drugiej karcie Object Inspectora znajdują się zdarzenia (Events). Każdy komponent posiada swój zbiór zdarzeń które możemy obsłużyć. Nazwa każdego zdarzenia składa się z przedrostka *On* i tekstu informującego czego dotyczy dane zdarzenie. Zdarzenia mają za zadanie sprawdzać czy nie wystąpiła określona czynność i w przypadku jej wystapienia wywołać odpowiadającą jej procedurę.

| | |
|---|---|
| OnClick | Po kliknięciu na dany komponent |
| Onactivate | Po "aktywowaniu", czyli kiedy komponent stanie się aktywny |
| OnClose | Po zamknięciu (formularza) |
| OnCreate | Po utworzeniu |
| Onshow | Po pokazaniu |
| OnDblClick | Po dwukrotnym kliknięciu |
| OnDeactivate | Odwrotność Onactivate |
| OnDestroy | Po "zniszczeniu" - kiedy użyjemy komendy free |
| OnKeyPress | Po naciśnięciu klawisza, kiedy komponent jest aktywny |
| OnMouseMove | Po ruszeniu myszą, kiedy kursor jest nad komponentem |
| OnPaint | Po najechaniu na przyciski formularza (minimize, maximize, close) |
| OnResize | Po zmianie wielkości |
| OnEnter | Po zaznaczeniu tab'em |
| OnExit | Po odznaczeniu tab'em |
| OnChange | Po zmianie, np. textu w komponencie TMemo |

Opis właściwości:

Właściwości pozwalają na zmianę położenia, rozmiaru, koloru, nazwy i wielu innych parametrów. Możemy je zmieniać bezpośrednio z poziomu Object Inspector-a w czasie projektowania aplikacji lub programowo za pomocą odpowiednich poleceń.
Przykładowe polecenie zmieniające właściwość *caption* komponentu Button1:

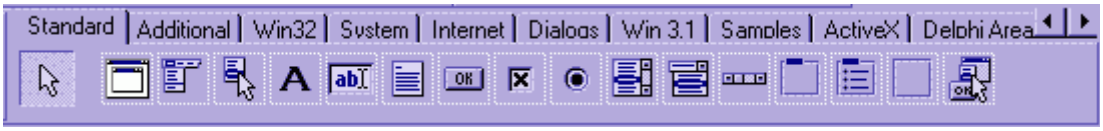
```
Button1.caption:='Nowy tekst nagłówka';
```

Skutek zmiany jakiejś właściwości jest natychmiastowy. W pewnych przypadkach niektóre właściwości są tylko do odczytu (read only), znaczy to że nie możemy danej właściwości zmienić. Możemy jednak ją odczytywać. Pełną listę właściwości znajdziemy w pomocy. Po zaznaczeniu komponentu i naciśnięciu klawisza F1 wyświetli się okienko z opisem dostępnych metod, właściwości i zdarzeń.

Oprócz właściwości i zdarzeń komponenty posiadają również *Metody*. Nie są one widoczne w Object Inspecto'rze. Metody są to funkcje i procedury które wykonuję na komponencie określone operacje.
Przykładowo metodę powodującą że przycisk staje się nie widoczny wywołujemy następującym poleceniem:

```
Button1.hide;
```

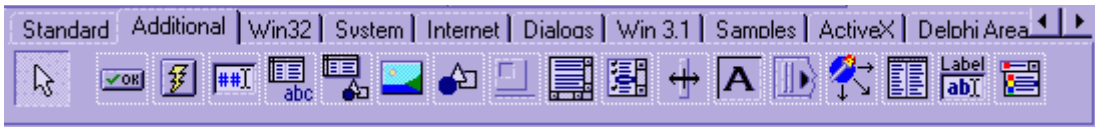
Karta *Standard*




Karta standart zawiera najczęściej wykorzystywane komponenty, służą one do projektowania wyglądu naszej aplikacji.

*Frames* - Ramki (frames) mają podobne właściwości jak formularze (forms) z tym wyjątkiem że ramka może być osadzona wewnątrz formy. Wprowadzenie ramek bardzo ułatwiło projektowanie wyglądu niektórych aplikacji.
*MainMenu* - główne menu danego formularza
*PopUpMenu* - menu wyświetlane po kliknięciu prawym przyciskiem myszki na danym obiekcie
*Label* - pole służące do wyświetlania tekstu
*Edit* - pole służace do edycji jednego wiersza tekstu
*Memo* - pole tekstowe z możliwością edycji, odczytu i zapisu wyświetlanego tekstu
*Button* - przycisk
*CheckBox* - pole wyboru
*RadioButton* - pole wyboru jedne z kilku opcji
*ListBox* - wyświetla listę elementów
*ComboBox* - wyświetla listę elementów z możliwością wpisania tekstu
*ScrollBar* - pasek przewijania
*GroupBox* - grupuje kilka komponentów np. typu RadioButton lub CheckBox
*RadioGroup* - grupuje komponenty RadioButton powodując że możliwe jest wybranie tylko jednego z nich
*Panel* - koleny komponent grupujący inne komponenty
*ActionList* - komponent pozwalający na dodawanie własnych procedur obsługi do niektórych akcji wykonywanych przez użytkownika

Karta *Additional*

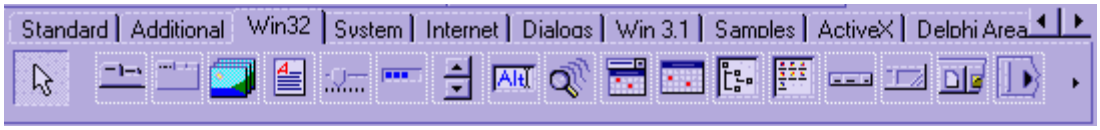



Karta additional zawiera uzupełniające komponenty kształtujące wygląd naszej aplikacji oraz ułatwiające komunikację z użytkownikiem.

*BitBtn* - przycisk na którym umieszczony jest rysunek
*SpeedButton* - przycisk umieszczeny w pasku narzędzi
*MaskEdit* - pole edycji pozwalające na filtrowanie i formatowanie danych wprowadzanych przez użytkownika
*StringGrid* - arkusz którego elementami są łańcuchy znaków
*DrawGrid* - arkusz przeznaczony do wprowadzania danych innych niż tekstowe
*Image* - komponent wyświetlający grafikę (także pliki JPEG)
*Shape* - figura geometryczna
*Bevel* - tworzy wypukłe lub wklęsłe linie, prostokąty, lub ramki
*ScrollBox* - przewijane okienko mogace zawierać inne komponenty

*CheckListBox* - przewijana lista z możliwością zaznaczenia poszczególnych pozycji
*Splitter* - służy do przesuwania części okienka
*StaticText* - komponent działający podobnie jak Label
*ControlBar* - pasek narzędzi z możliwością przestawiania poszczególnych pozycji
*ApplicationEvents* - niewizualny komponent umożliwiający obsługe globalnych zdarzeń aplikacji
*ValueListEditor* - edytor listy wartości
*LabeledEdit* - pole edycyjne z tekstem opisu
*ColorBox* - lista wyboru kolorów systemowych

Karta *Win32*




Zawarte tu komponenty wprowadzają nowe elementy sterujące dając dostęp do 32-bitowego interfejsu systemu Windows.

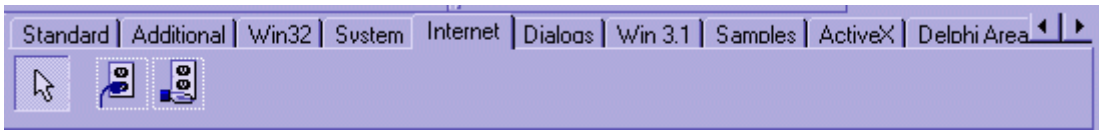
*TabControl* - umożliwia tworzenie zakładek
*PageControl* - składa się z wielu kart między którymi przechodzic można za pomocą zakładek
*ImageList* - służy do przechowywania kilku elementów graficznych
*RichEdit* - pole edycyjne z dostępnym formatowaniem tekstu (różne czcinki, kolory, atrybuty)
*TrackBar* - pasek przewijania typu suwak
*ProgressBar* - wskaźnik postepu
*UpDown* - komponent związany z jakąś wartością pozwalający na jej zwiększenie bądź zmniejszenie
*HotKey* - tworzy klawisze szybkiego dostępu
*Animate* - komponent pozwalający na odtwarzanie plików AVI (wyłącznie nieskompresowane lub z kompresją RLE), zmiane parametrów otwarzania a także odtwarzanie animacji sytemowych (np. kopiwanie pliku)
*DateTimePicker* - kalendarz pozwalający na wybranie dowolnej daty z przyszłości lub przeszłości
*MonthCalendar* - kalendarz wyświetlający okienko miesięczne
*TreeView* - pozwala na wyświetlanie elementów w postaci drzewa
*ListView* - pozwala na wyswietlenie elementów składających się z ikony i etykiety
*HeaderControl* - umożliwia tworzenie nagłówka składającego się z wielu sekcji
*StatusBar* - linia statusu formularza
*ToolBar* - pasek narzędzi przeznaczony do umieszczania na komponencie ControlBar
*CoolBar* - bardziej zaawansowana wersja komponentu ControlBar
*PageScroller* - pozwala na przesuwanie okienka w jednej płaszczyźnie (np. pasek narzędziwy nie mieszczący się na formularzu)
*ComboBoxEx* - rozwijana lista której pozycje mogą zawierać obrazki oraz regulowane wcięcia

Karta *System*


Na tej karcie znajdują się komponenty korzystające bezpośrednio z funkcji systemowych .

*Timer* - wywołuje zadaną procedurę w określonych odstępach czasu
*PaintBox* - obszar przeznaczony do wykonywania na nim operacji graficznych
*MediaPlayer* - odtwarzacz multimedialny
*OleContainer* - przeznaczony do osadzania w naszym programie obiektów OLE
*DdeClientConv, DdeClientItem, DdeServerConv, DdeServerItem* - zestaw komponentów służących do przekazywania danych między aplikacjami

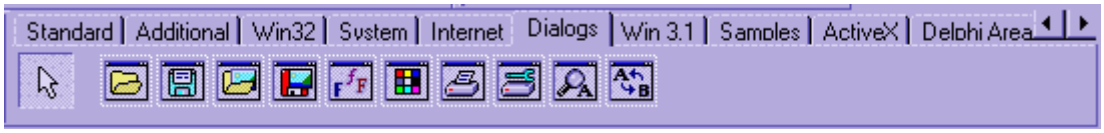
Karta *Internet*




Karta zawierająca komponenty do pracy w Internecie. W D6 Personal została drastycznie okrojona.

*ClientSocket* - umożliwia komunikację z serwerem za pomocą TCP/IP
*ServerSocket* - komponent dla serwera obsługujący TCP/IP

Karta *Dialogs*



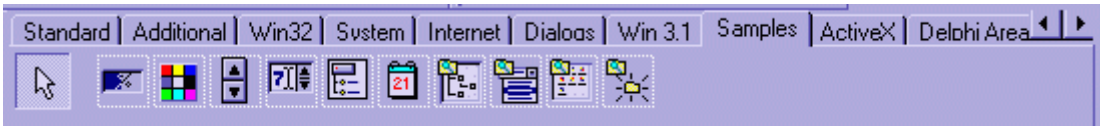

Zawiera komponenty wywołujące różnego typu okienka dialogowe.

*OpenDialog* - okienko otwierania pliku
*SaveDialog* - okienko zapisywania pliku
*OpenPictureDialog* - okienko otwierania pliku z podglądem graficznym
*SavePictureDialog* - okienko zapisywania pliku z podglądem graficznym
*FontDialog* - okienko wyboru czcionki
*ColorDialog* - okienko wyboru koloru
*PrintDialog* - okienko drukowania
*PrinterSetupDialog* - okienko ustawień drukarki
*FindDialog* - okienko obsługujące procedury przeszukiwania
*ReplaceDialog* - okienko obsługujące procedury zamiany zadanej frazy

Karta *Win 3.1*

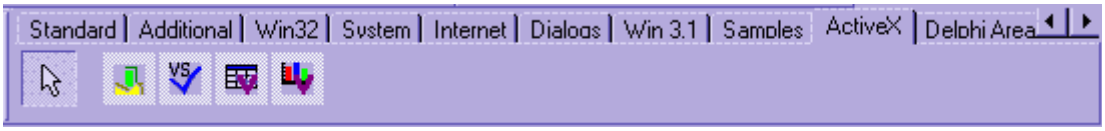
Na tej karcie znajdują się komponenty pochodzące z Delphi 1.0 . Wykorzystywane były w Windows 3.1 . Większość z nich ma swoje odpowiedniki w komponentach na karcie Win32. Zostały umieszczone wyłącznie w celu zachowania wstecznej kompatybilności.

Karta *Samples*




Na tej karcie znajdują się przykładowe komponenty stworzone z użyciem innych komponentów. Są to komponenty wykonujące bardzo konkretne zadania, przydające się jedynie w szczególnych przypadkach.

Karta *ActiveX*




Na tej karcie znajdują się przykładowe kontrolki ActiveX. Nie wchodzą one w skład biblioteki VCL. Jeżeli zamieścisz je w swojej aplikacji będziesz zmuszony do do dostarczenia ich w osobnym pliku. Możliwe jest także użycie kontrolek systemowych (np. Windows MediaPlayer) lub udostepnionych przez inne programy (np. MS Office). Opis instalacji dodatkowych kontrolek znajdzie się w dalszej części kursu.