Delphi#2

1. Czym jest Delphi?

1.1. Narzędziem do szybkiego tworzenia aplikacji (RAD - *Rapid Application Development*).
1.2. Delphi jest językiem programowania.
1.3. Pakiet Embarcadero Delphi XE2 jest płatny.
1.4. Środowiskiem wzorowanym na Delphi i obecnie rozwijanym jako wolne oprogramowanie jest zintegrowane środowisko programowania o nazwie Lazarus (http://www.lazarus.freepascal.org/).

2. Uruchomienie środowiska Delphi XE2

Menu Start->Embarcadero RAD Studio XE2->Delphi XE2

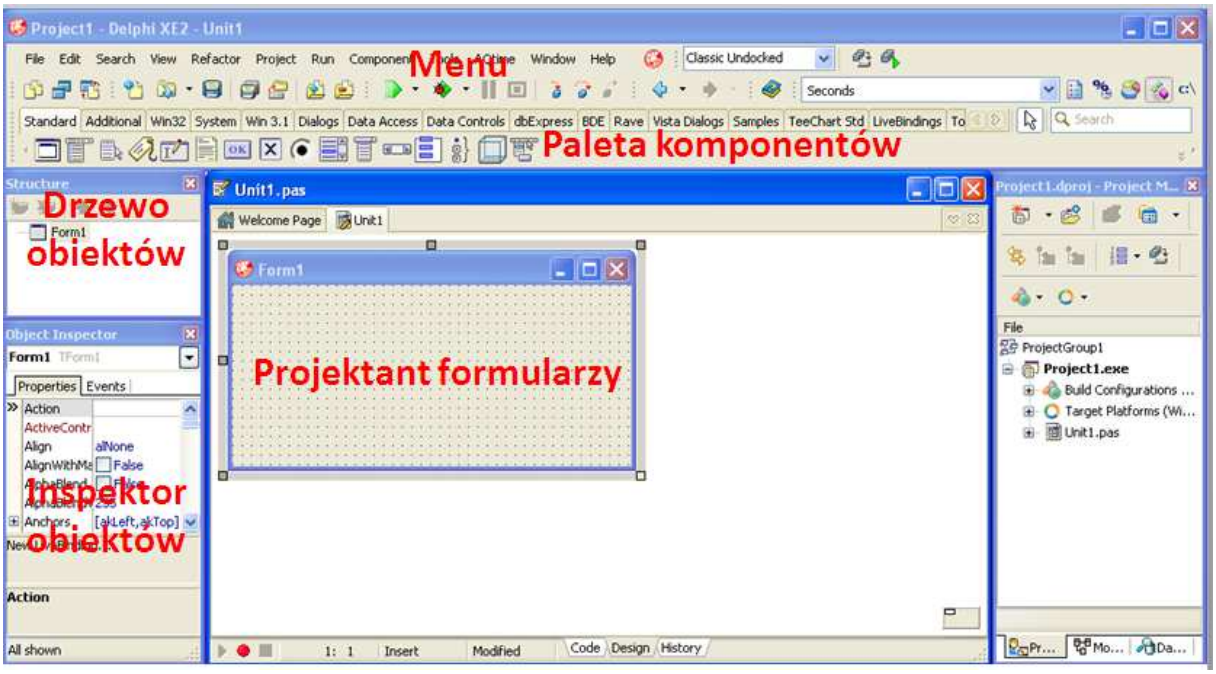
3. Kompilacja i uruchamianie programu

Kompilacja projektu następuje poprzez wybranie z menu *Project* polecenia *Compile* (*Ctrl+F9*).
Uruchomienie programu następuje poprzez wciśnięcie klawisza *F9* albo przez wybranie z menu *Run* polecenia *Run*.

4. Wygląd Delhi XE2

Tak wygląda Delphi po uruchomieniu:

Należy zmienić *Default Layout* na *Classic Undocked* – widoczna staje się paleta komponentów i można dostosować wygląd środowiska do swoich potrzeb:



Paski narzędzi (*toolbars*)

Pozwalają na szybki dostęp do często wykonywanych operacji i używanych poleceń poprzez jednokrotne kliknięcie odpowiedniego przycisku. Zazwyczaj z każdym ze wspomnianych przycisków skojarzona jest podpowiedź, zawierająca krótki opis funkcji związanej z danym przyciskiem.
Poniżej na rysunku są 3 paski narzędziowe z opisem (z 6 dostępnych).
W przypadku paska debugowania – *trace into* oznacza pracę krokową, linijka po linijce z wchodzeniem do wnętrza funkcji i procedur skrót *F7*, natomiast *step over* oznacza pracę krokową, bez wchodzenia do wnętrza funkcji i procedur skrót *F8*.

Paleta komponentów (*The Component palette*)

Jest to okno podzielone na zakładki, grupujące komponenty (obiekty, gotowe „klocki” np. przyciski, pola edycyjne) tematycznie.

Projektant formularzy (*The Form Designer*)

Podgląd tworzonej aplikacji. Formularze odpowiedzialne są za ostateczny wygląd aplikacji. Rozmieszczenie i rozmiar komponentów można korygować za pomocą myszy, a do ewentualnej modyfikacji ich wyglądu i zachowania służą: inspektor obiektów oraz edytor kodu.
Pomocnicza siateczka ułatwia nanoszenie komponentów, znika po uruchomieniu programu przez *F9.*

Inspektor obiektów (*Object Inspector*)

Pokazuje listę właściwości (wyświetlone w porządku alfabetycznym) oraz listę zdarzeń skojarzonych z danym komponentem; skrót *F11*. Umożliwia modyfikację właściwości komponentów oraz zmianę ich sposobu reagowania na zdarzenia. Właściwościami obiektu są dane określające na przykład jego wysokość, szerokość etc. Zdarzenie związane jest natomiast z wystąpieniem pewnej sytuacji, np. naciśnięciem klawisza.

Edytor kodu (*Code editor*)

Tutaj wpisuje się cały szereg poleceń czyli kod naszego programu.
Każdorazowe włączenie do aplikacji nowego formularza powoduje automatyczne wygenerowanie nowego modułu kodu. Możliwe jest również dodawanie modułów kodu nie reprezentujących formularzy.
Przydatne skróty:
Ctrl+spacja – wymuszenie wyświetlenia listy podpowiedzi (dokańczanie kodu).
Ctrl+J – wyświetlenie listy szablonów kodu.
Ctrl+O+U – zmienia wielkość liter w zaznaczonej części kodu.
Ctrl+K+E – zmienia wszystkie litery na małe.
Ctrl+K+F – zmienia wszystkie litery na duże.
F1 – wyświetlenie pomocy.
F12 – przełączanie edytor kodu/formatka (projektant formularzy).

5. Pliki projektu

Po kompilacji programu w katalogu projektu znajdować mogą się następujące pliki:

- *.pas* - pliki źródłowe; w nich znajdują się kody źródłowe formularzy lub modułów (związanych z formularzem lub samodzielnych).
- *.dfm* - zawiera informacje dotyczące komponentów umieszczonych na formularzu (dane o ich położeniu, nazwie itp.).
- *.dcu* - skompilowany plik *.pas; plik ten nie jest potrzebny - po kolejnej próbie kompilacji Delphi odtworzy go na podstawie kodów źródłowych.

- *.dpr* - główny plik programu, tzw. plik projektowy (programowy).
- *.res* - tzw. zasoby. Plik ten zawiera obrazy bitmapowe, kursory, łańcuchy etc.
- *.dof* – plik z opcjami projektu, który jest plikiem tekstowym z informacjami o ustaleniach dotyczących kompilatora, programu łączącego, katalogów, dyrektyw kompilatora i parametrów uruchomienia programu.
- *.dsk* – plik z ustaleniami środowiska, który umożliwia odtworzenie środowiska w takiej postaci, w jakiej nastąpiło zapisanie projektu.
- *.dpk* – pliki z tekstami źródłowymi pakietów.

Z każdym plikiem formatki (formularza, formy) skojarzony jest 1 plik (.pas), który ma domyślnie taką samą nazwę.

6. Prosta aplikacja

6.1. Zmiana właściwości *Caption* (tekst w pasku głównym programu) w *Inspektorze obiektów* dla formatki.

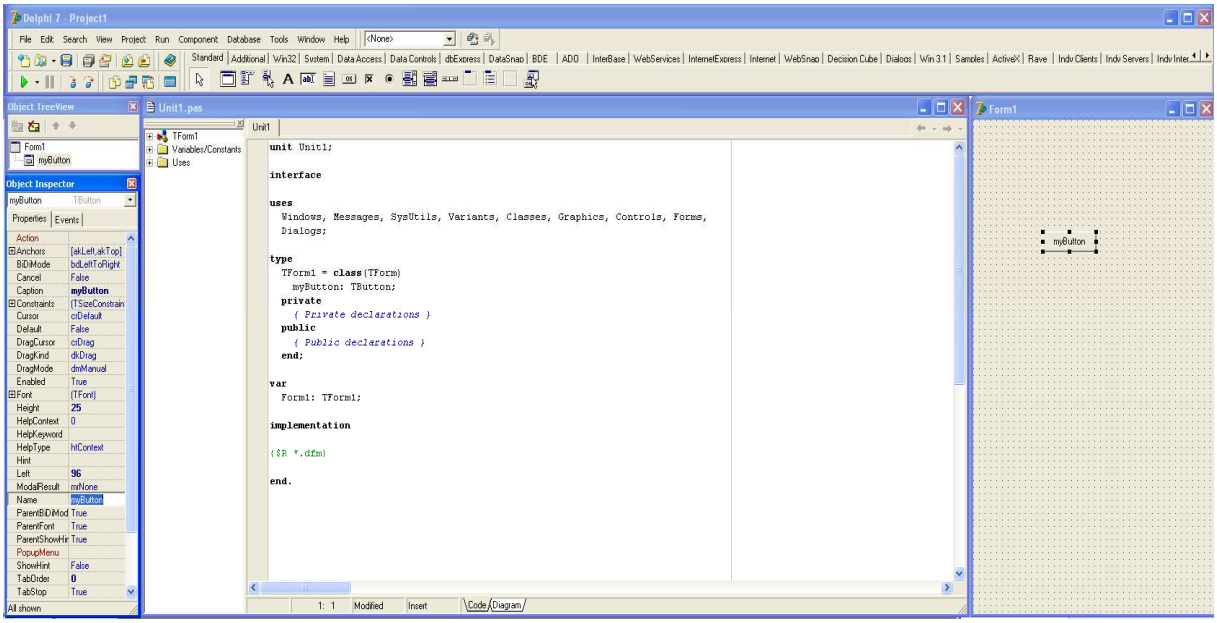
6.2. Zapisz projekt.

File->Save Project As

Utwórz nowy katalog dla projektu. Nazwa dla pliku z kodem źródłowym: *Hellof.pas*, nazwa dla projektu *Hello.dpr*

6.3. Dodaj przycisk

Kliknij na palecie komponentów w miejscu obrazka z przyciskiem. Przesuń kursor myszy nad formatkę, lewym przyciskiem myszy kliknij na obszarze formatki, w miejscu gdzie chcesz umieścić przycisk. W *Inspektorze obiektów* zmień nazwę przycisku np. na *myButton* oraz zmień etykietę przycisku *Caption*.



6.4. Oprogramuj zdarzenie kliknięcia w obrębie przycisku.

Podwójnie kliknij przycisk umieszczony na formatce. Delphi automatycznie przeniesie nas do *Edytora kodu* i ustawi kursor wewnątrz procedury.
To co ma zostać wykonane po wciśnięciu przycisku należy wpisać pomiędzy słowa *begin* i *end*, np.

ShowMessage('Hello World'); - to polecenie powoduje wyświetlenie okienka informacyjnego Windows. Tekst zawarty w apostrofach zostanie zawarty w oknie.

MessageDlg(tekst, ikona, przycisk, pomoc); - funkcja służąca do wyświetlania komunikatów w okienku dialogowym.

Tekst – tekst wpisany pomiędzy znaki "
Ikona – ikona widoczna w oknie komunikatu
Przycisk – rodzaj przycisku w oknie ([] – oznacza zbiór; notację należy stosować, nawet jeśli wybierany jest tylko 1 przycisk)
Pomoc - numer tematu pomocy

Dostępne ikony:
mtWarning - ostrzeżenie
mtError - błąd
mtInformation - informacja
mtConfirmation – potwierdzenie

Przykładowe, dostępne klawisze:
mbOK
mbYes
mbNo
mbCancel
mbHelp

Przykład wywołania:

MessageDlg('Hello World', mtInformation, [mbOK], 0);

```
procedure TForm1.myButtonClick(Sender: TObject);
begin
      ShowMessage('Hello World');
end;

end.
```

8. Środowisko Delhi

8.1. Komponenty

Komponent jest samodzielnym elementem, obiektem wykonującym określoną predefiniowaną funkcję. Komponentem jest np. etykieta, pole edycji, przycisk czy lista wyboru. Patrząc na komponenty ze strony projektanta aplikacji, mamy do czynienia z wygodnymi i łatwymi w obsłudze narzędziami. Publiczny interfejs komponentów biblioteki VCL (*Visual Component Library*) tworzą trzy składniki: właściwości, metody i zdarzenia.

Właściwości są elementami komponentu odpowiadającymi za jego zachowanie. Wiele komponentów ma wspólne właściwości. Na przykład, wszystkie komponenty wizualne posiadają właściwości `Top` oraz `Left`. Określają one miejsce na formularzu, w którym zostanie umieszczony komponent – zarówno w fazie projektowania jak i wykonywania programu. W celu obejrzenia właściwości danego komponentu kliknij na zakładce Inspektora Obiektów `Properties`.

Metody komponentów VCL są funkcjami i procedurami, które są wywołane po to, aby komponent wykonał pewną akcję. Wszystkie komponenty wizualne na przykład mają metodę `Show` wyświetlającą komponent, oraz metodę `Hide` ukrywającą komponent. Metody mogą być zadeklarowane jako publiczne (`public`), chronione (`protected`) albo prywatne (`private`). Użytkownicy komponentu mogą wywoływać jedynie metody publiczne.

Zdarzenie definiuje się jako wszystko to, co zdarzy się w komponencie, a co może być ważne dla użytkownika. Każdy komponent jest tak projektowany, aby mógł odpowiadać na pewne zdarzenia (najczęściej jest to zdarzenie systemu Windows) np. aktywacje menu, naciśnięcie przycisku czy przesunięcie okienka. Zdarzenia są obsługiwane przez metody zwane procedurami obsługi zdarzeń. Nazwy zdarzeń odpowiadają opisowi zdarzenia, na które reagują. Na przykład, zdarzenie obsługi

kliknięcie myszą posiada nazwę `OnClick`. W celu obejrzenia listy zdarzeń, które może obsłużyć dany komponent kliknij na zakładce Inspektora Obiektów `Events`.

8.1.1. Przycisk (`Button`)

Standardowy komponent `Button` posiada domyślną wysokość (`Height`) równą 25 pikseli i domyślną szerokość (`Width`) równą 75 pikseli. Zazwyczaj przycisk umieszcza się na formularzu, oprogramowuje jego zdarzenie `OnClick` – i na tym koniec.

Przykład

Patrz punkt **7. Prosta aplikacja**.

8.1.2. Etykieta (`Label`)

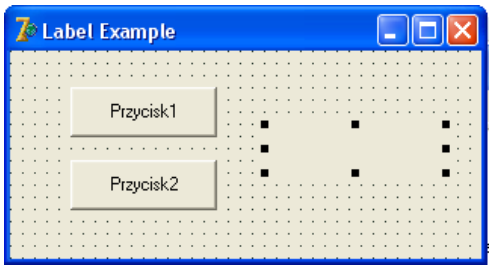
Komponent `Label` służy do wyświetlania tekstu w formularzu. Tekst ten może być określany w fazie projektowania i później nie ulegać już zmianom. W innych przypadkach, etykieta może zachowywać się w sposób dynamiczny, ulegając zmianom w trakcie pracy, w sposób zależny od przebiegu programu. Do zmiany tekstu etykiety w trakcie pracy programu służy właściwość `Caption`. Komponent `Label` nie posiada żadnych specjalizowanych metod poza tymi, którymi dysponują też inne komponenty. Niektóre właściwości specyficzne dla komponentu `Label` zostały przedstawione w tabeli poniżej.

Właściwość	Opis
`AutoSize`	Przy ustawionej wartości `True` etykieta automatycznie zmienia swój rozmiar dopasowując się do tekstu zawartego we właściwości `Caption`. Jeżeli wartością jest `False`, tekst wychodzący poza prawą krawędź jest obcinany. Wartością domyślną jest: `True`.
`FocusControl`	Etykieta jest komponentem nieokienkowym, stąd też nie może więc przyjąć stanu aktywności i nie można przejść do niej przy użyciu klawisza `Tab`. Czasami etykieta służy jako tekst dla kontrolki takiej jak np. pole edycji. Wtedy etykiecie można przypisać klawisz akceleratora (używając znaku `&`), a jej właściwości `FocusChange` przypisać komponent, do którego ma być kierowany stan aktywności, gdy klawisz akceleratora zostanie naciśnięty.
`Transparent`	Kiedy jest ustawiona na `True` ignorowana jest właściwość `Color` i wszystko, co znajduje się pod etykietą staje się przezroczyste. Właściwość ta jest przydatna na przykład przy umieszczaniu etykiet na tle stworzonym z bitmapy. Wartością domyślną jest: `False`.
`WordWrap`	Kiedy jest ustawiona na `True` tekst we wnętrzu etykiety będzie zawijany do nowej linii, gdy osiągnie jej prawą krawędź. Wartością domyślną jest: `False`.

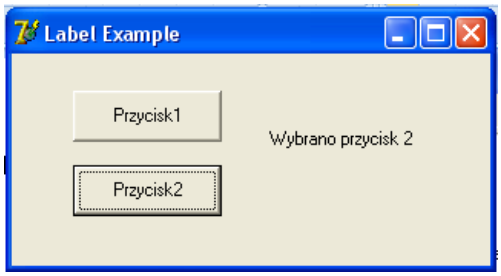
Ciekawym i przydatnym typem etykiety jest komponent `StaticText`. Rożni się on tym od zwykłego komponentu `Label`, że jest komponentem okienkowym (skojarzony jest z nim uchwyt okna). Niektóre pożyteczne operacje z użyciem Win32 API odnoszą się do konkretnego okna i jako takie wymagają właśnie uchwytu – którego komponent `Label` nie posiada. Komponent `StaticText` posiada specjalną wartość `sbsSunken` dla właściwości `Border` z efektem wklęsłości (standardowa etykieta tego nie posiada).

Przykład

Utwórz aplikację zawierającą 2 przyciski (komponent `Button`) oraz etykietę (komponent `Label`), jak pokazano na rysunku poniżej:



Po naciśnięciu każdego z przycisków na etykiecie powinien ukazać się napis wskazujący, który przycisk został naciśnięty:



W celu wykonania przykładu należy najpierw wstawić 1 etykietę (komponent `Label`) oraz 2 przyciski (komponent `Button`) do formularza. Należy ustawić właściwość `Caption` przycisków na `Przycisk1`, `Przycisk2`.

Implementacja procedur `OnClick` dla przycisków:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Label1.Caption := 'Wybrano przycisk 1';
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
    Label1.Caption := 'Wybrano przycisk 2';
```

```
end;
```

Kod źródłowy przykładu:

http://www.cs.put.poznan.pl/arybarczyk/cw1/label.zip

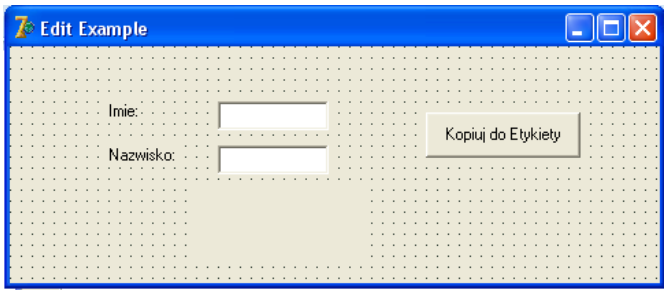
8.1.3. Pole tekstowe (`Edit`)

Komponent `Edit` jest prostą kontrolką edycji jednowierszowej. Komponent ten nie posiada właściwości `Align` oraz `Alignment`. W kontrolce pojedynczej linii edycji tekst może być wyrównywany jedynie do lewej strony, stąd brak właściwości `Alignment`. Właściwość `Align` nie występuje w komponencie `Edit`, ponieważ nie powinien on mieć możliwości powiększenia się do pełnego obszaru klienta w oknie.

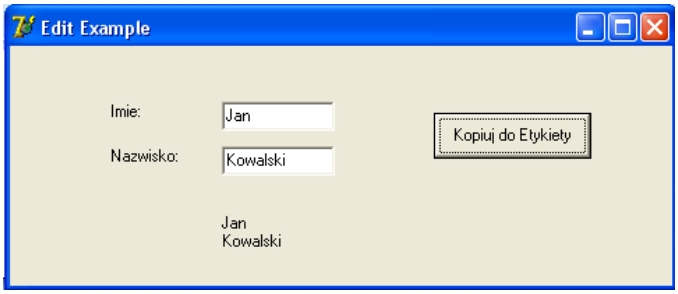
Jeżeli potrzebny jest komponent edycji z wyrównywaniem do prawej strony lub centrowaniem, należy użyć komponentu `Memo` ustawiając jego wysokość na wymiar odpowiadający wysokości standardowego komponentu `Edit`. Następnie należy zmodyfikować właściwość `Alignment` według własnych wymagań.

Przykład

Utwórz 2 pola tekstowe (komponenty `Edit`), 2 etykiety (komponenty `Label`) opisujące te pola, 1 etykietę (komponent `Label`) poniżej, do której zostanie skopiowana zawartość pól tekstowych po wciśnięciu przycisku oraz 1 przycisk (komponent `Button`), jak pokazano na rysunku poniżej:



Przycisk powinien po kliknięciu kopiować zawartość pól tekstowych do etykiety.



W celu wykonania przykładu należy najpierw wstawić 2 pola tekstowe (komponent `Edit`), 3 etykiety (komponenty `Label`) oraz 1 przycisk (komponent `Button`) do formularza.

Implementacja procedury `OnClick` dla przycisku „Kopiuj do Etykiety”:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  //jesli w polu tekstowym Edit1 i Edit2 zostal wpisany jakis tekst
  if (Edit1.Text <> '') and (Edit2.Text <> '') then
    //kopiuj tekst z pol tekstowych do wlasciwosci Caption etykiety
    Label3.Caption := Edit1.Text + #13 + Edit2.Text;
end;
```

Uruchom aplikację i sprawdź efekty jej działania.

Kod źródłowy przykładu: http://www.cs.put.poznan.pl/arybarczyk/cw1/edit.zip

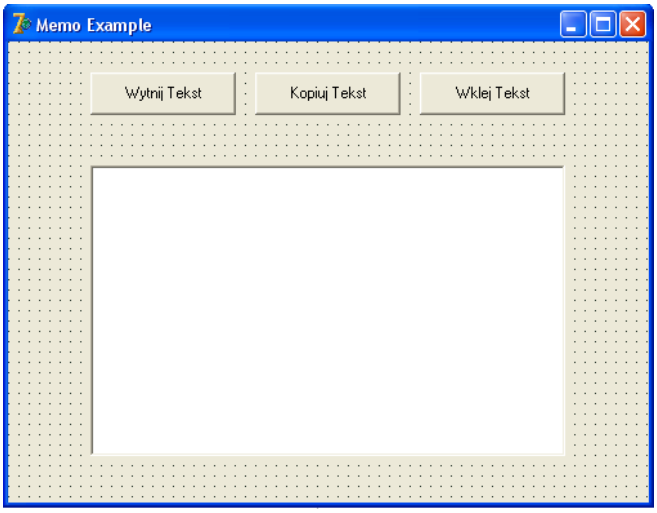
8.1.4. Pole tekstowe (`Memo`)

Komponent `Memo` realizuje wielowierszową kontrolkę edycyjną. Najbardziej znaczącą właściwością tego komponentu jest `Lines`, która umożliwia zapisywanie zawartości komponentu `Memo` na dysk, ładowanie do komponentu tekstu pochodzącego z pliku lub przetwarzanie każdej linii notatki w sposób indywidualny.

Cechą szczególną komponentu `Memo` jest również jego właściwość `ScrollBars`. Określa ona, czy komponent posiada poziomy, pionowy czy też obydwa paski przewijania.

Przykład

Utwórz pole tekstowe (komponent `Memo`) oraz 3 przyciski (komponenty `Button`), jak pokazano na rysunku poniżej:



Aplikacja wyświetla okno do wpisywania tekstu i tworzy pasek z przyciskami. Przyciski paska umożliwiają kopiowanie, wycinanie oraz wklejanie tekstu.

Do operacji na tekście wykorzystano podstawowe metody komponentu `Memo`:
`CutToClipboard` – metoda wycinająca wyselekcjonowany tekst do schowka;
`CopyToClipboard` - metoda kopiująca wyselekcjonowany tekst do schowka;
`PasteFromClipboard` - metoda wklejająca wyselekcjonowany tekst ze schowka w miejscu położenia kursora;
Inne ciekawe metody i właściwości:
`ClearSelection` - metoda wycinająca wyselekcjonowany tekst;
`Clear` - metoda czyszcząca zawartość pola `Memo`;
`Lines` – właściwość, lista łańcuchów przedstawiających wiersze tekstu, ich właściwości i metody. Dwie podstawowe metody tej właściwości to:

`Lines.LoadFromFile` - wczytywanie do pola `Memo` zawartości pliku o nazwie określonej parametrem `nazwa`.

`Lines.SaveToFile` - zapisywanie zawartosci pola `Memo` do pliku o nazwie określonej parametrem `nazwa`.

W celu wykonania przykładu należy najpierw wstawić pole tekstowe (komponent `Memo`) oraz 3 przyciski (komponenty `Button`) do formularza. Następnie we właściwościach `Memo` należy dwukrotnie kliknąć w polu właściwości `Lines`, otworzy się `StringListEditor`. W edytorze należy usunąć domyślnie wpisany tekst „Memo1” i zatwierdzić przyciskiem OK.
Implementacja procedur `OnClick` dla przycisków:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  Memo1.CutToClipboard;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
  Memo1.CopyToClipboard;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
  Memo1.PasteFromClipboard;
end;
```

Uruchom aplikację i sprawdź efekty jej działania.

Kod źródłowy przykładu:

http://www.cs.put.poznan.pl/arybarczyk/cw1/memo.zip

8.1.5. Lista wyboru (`ListBox`)

Komponent `ListBox` stanowi standardową listę wyboru w Windows. Jeżeli lista zawiera więcej elementów aniżeli jest w stanie jednocześnie wyświetlić, dostęp do pozostałych jest możliwy dzięki paskom przewijania. Elementy listy można sortować poprzez ustawienie właściwości `Sorted` na `True`. Numer elementu aktualnie zaznaczonego znajduje się we właściwości `ItemIndex` (liczony od 0). Jeśli natomiast chcielibyśmy odczytać jaki element znajduje się pod tym numerem należy użyć tablicy elementów listy `Items` np.

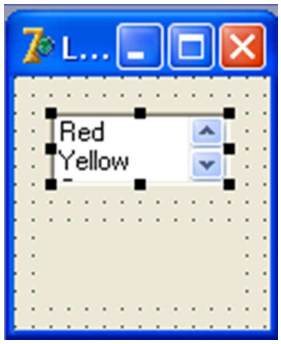
```
ListBox1.Items.Strings[ListBox1.ItemIndex].
```

`Items.Strings[nr]` określa element z listy o indeksie `nr` (w postaci łańcucha), `Text` zwraca zawartość listy w postaci pojedynczego łańcucha.

Należy pamiętać, że w przypadku komponentu `ListBox` po usunięciu elementu z listy numeracja pozostałych elementów na liście `Item` może ulec zmianie.

Przykład

Utwórz listę wyboru `ListBox` z nazwami kolorów oraz etykietę jak na rysunku poniżej:



Po wybraniu koloru z listy etykieta zostanie pomalowana wybranym kolorem oraz nazwa koloru zostanie wyświetlona na etykiecie.

W celu wykonania przykładu należy najpierw wstawić etykietę (komponent `Label`) i listę (komponent `ListBox`) do formularza.

Następnie we właściwościach `ListBox` należy dwukrotnie kliknąć w polu właściwości `Items`, otworzy się `StringListEditor`. W edytorze wpisać w osobnych wierszach kolory (będą to elementy listy wyboru) np.

```
Red
Yellow
Green
```

a następnie zatwierdzić przyciskiem OK.

Implementacja procedury `OnClick` dla komponentu `ListBox`:

```
procedure TForm1.ListBox1Click(Sender: TObject);
begin
  //indeks elementu wybranego z listy
  case ListBox1.ItemIndex of
    0:Label1.Color := clRed;
    1:Label1.Color := clYellow;
    2:Label1.Color := clGreen;
  end;
  Label1.Caption := ListBox1.Items.Strings[ListBox1.ItemIndex];
end;
```

Uruchom aplikację i sprawdź efekty jej działania.

Kod źródłowy przykładu:

http://www.cs.put.poznan.pl/arybarczyk/cw1/lista.zip

9. Materiały

[1]. "Delphi 4 dla każdego" Kent Reisdorph
[2]. Delphi Language Guide
[3]. http://4programmers.net/
[4]. http://www.marcocantu.com/ (Essential Pascal, Essential Delphi)
[5]. „Delphi 6. Vademecum profesjonalisty. Tom 1." Xavier Pacheco, Steve Teixeira
[6]. „Język programowania Delphi" Andrzej Marciniak

Dodatkowe strony:

http://delphi.cartall.com.pl/
http://delphi.icm.edu.pl/
http://cpw.net.pl/