Zestaw ćwiczeń C++ Builder – podstawowe komponenty

Opracowanie - Hanna Stachera

Każde zadanie zapisujemy w oddzielnym katalogu, przed kompilacją zapisując Unit i Projekt z nazwą np. **zad_n.**

Kompilujemy program poleceniem Run, lub F9.

Spis treści

C++ Builder
Ćwiczenia

Ćwiczenie 1. Nazwa i ikona dla aplikacji

Z menu głównego wybieramy **Project/Options** zakładka **Application** w oknie **Title** wpisujemy nazwę aplikacji np. Kalkulator, a przez przycisk **LoadIcon** wybieramy ikonę dla naszej aplikacji.

Ćwiczenie 2. Przyciski BitBtn

1. Wstawimy na formularz z zakładki **Additional** 5 przycisków BitBtn.
2. Dla każdego z nich wybierzmy inną właściwość **Kind** np. dla pierwszego wybierzmy: **bkAbort**.
3. Przygotujmy plik tlo.bmp i zapiszmy go w katalogu zad_2 (tam gdzie pliki aplikacji).
4. Umieśćmy na formularzu kolejny przycisk **BitBtn**, wywołajmy jego właściwość **Glyph** i wprowadźmy do niej przez wyświetlony edytor przygotowany wcześniej plik tlo.bmp
5. Ustawiamy odpowiedni rozmiar przycisku.
6. We właściwości **Caption** możemy wprowadzić tekst opisujący przycisk.
7. Właściwością **NumGlyphs** możemy regulować wielkość obrazka na przycisku.
8. Jeśli chcemy wyświetlić podpowiedź dla przycisku wprowadźmy jej treść do właściwości **Hint**, a następnie ustawiamy **Showhint** na true.
9. Jeśli chcemy zablokować przycisk ustawiamy właściwość **Enabled** na false.
10. Jeśli chcemy zmienić sposób wyświetlania kursora po najechaniu myszą na przycisk wybierzmy z listy odpowiednie ustawienie przy właściwości **Cursor**.
11. Sposób wyświetlania tekstu na przycisku możemy zmienić przez właściwość **Font**.
12. Właściwością **Layout** możemy regulować ustawienie ikonki w stosunku do tekstu opisującego przycisk: góra, dół, prawo, lewo.
13. Jeśli chcemy przycisk schować - ustawiamy **Visible** na false.

Ćwiczenie 3. Przycisk Button - wyświetlanie i ukrywanie komponentów

1. W Formularz **Caption** wpiszmy tytuł formularza.
2. Umieśćmy na formularzu komponent **Image** z karty Additional.
3. Za pomocą właściwości **Picture** wprowadźmy do komponentu odpowiedni obrazek np. **z /Program Files/Common Files/Borland Shared/Images/Splash/256Color**
4. Rozciągnijmy odpowiednio pole **Image.**
5. Umieśćmy na formularzu przycisk **Button** z karty Standard,
   a. ustawiamy **Caption** Pokaż\Ukryj
   b. w Events – **OnClick** wprowadźmy kod:

```
void __fastcall TForm::ButtonClick(TObject *Sender)
{
static bool isVisible;
isVisible = !isVisible;
if (isVisible) Image1->Hide();
else Image1->Show();
}
```

Jeśli chcemy aby obrazek pokrywał cały obszar roboczy formularza zmieńmy jego właściwość **Align** na **alClient**.

Ćwiczenie 4. Okno dialogowe - zmiana koloru formularza, ukrywanie i wyświetlanie przycisku

1. Wybieramy kartę Events dla formularza.
2. W pozycji **OnActivate** wpisujemy
   MessageBox (0, "C++Builder", „Kolory!",0);
3. Z karty Standard wybieramy i wstawiamy do formularza przycisk **Button**.
4. Nadajemy mu nazwę w **Caption** np. „**Zmień kolor**".
5. Przypisujemy mu reakcję na kliknięcie **OnClick**
   Form1->Color=clGreen;
6. Modyfikujemy funkcję **OnClick** formularza:

```
if (Button1->Visible==true)
Button1->Hide();
else
Button1->Show();
```

Jeśli chcemy zobaczyć jak zmieni się przycisk w oknie dialogowym zmieńmy wartość ostatniego parametru funkcji **MessageBox** (z 0) odpowiadającego za ilość i jakość przycisków w oknie dialogowym np. na 1.

Ćwiczenie 5. Canvas – rysowanie prostokątów

1. Ustawiamy **Color** formularza np. na „clWhite”.
2. Do zdarzenia **„onClick”** Formularza wprowadźmy kod:
   Canvas->TextOut(350, 220, "Rysowanie na ekranie");
3. Do zdarzenia **„dblCkick”** Formularza wprowadźmy kod:
   Canvas->Brush->Color = clBlue;
   Canvas->Pen->Color = clRed;
   Canvas->Pen->Width = 30;
   Canvas->Rectangle(30, 30, 100, 100);

Ćwiczenie 6. Canvas – rysowanie figur geometrycznych, wykorzystanie komponentu Image

1. Ustawiamy **„Color”** formularza np. na „**clWhite**”.
2. Wprowadźmy 2 komponenty **„Image”** z karty **Additional.**
3. Umieśćmy pod wprowadzonymi elementami trzy przyciski **„Button”**.
4. Nazwijmy przyciski „Rys.1”, „Rys.2”, „**Czyszczenie**” (**w Caption**).
5. Do kodu związanego ze zdarzeniem **„onClick”** przycisku „Rys.1” wpiszmy:
   Image1->Canvas->Brush->Color = clBlack;
   Image1->Canvas->Brush->Style = bsDiagCross;
   Image1->Canvas->Ellipse(0, 0, Image1->Width, Image1->Height);
6. Do kodu związanego ze zdarzeniem **„onClick”** przycisku „Rys.2” wpiszmy:
   Image2->Canvas->Brush->Color = clBlack;
   Image2->Canvas->Brush->Style = bsDiagCross;
   Image2->Canvas->Rectangle(0, 0, Image2->Width, Image2->Height);
7. Do kodu związanego ze zdarzeniem **„onClick”** przycisku „Czyszczenie” wpiszmy:
   Image1->Canvas->Brush->Color = clWhite;
   Image1->Canvas->Brush->Style = bsDiagCross;
   Image1->Canvas->Ellipse(0, 0, Image1->Width, Image1->Height);
   Image2->Canvas->Brush->Color = clWhite;
   Image2->Canvas->Brush->Style = bsDiagCross;
   Image2->Canvas->Rectangle(0, 0, Image2->Width, Image2->Height);

Ćwiczenie 7. Canvas – zastosowanie właściwości Width i Height dla Tpen, losowe rysowanie figur, wykorzystanie zegara TTimer oraz funkcji Random

1. Do formularza wstawmy komponent **„Timer”** z karty System.
2. Ustawiamy właściwość **„OnActivate”** dla Formularza wpisując kod:

```
void __fastcall TForm1::FormActivate(TObject *Sender)
{
 Timer1->Interval = 70;
 randomize();
}
```

3. Ustawiamy zdarzenie **“OnTimer”** dla komponentu **Timer** wpisując następujący kod:

```
int x, y;
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
 x = random(Screen->Width - 10);
 y = random(Screen->Height - 10);
 Canvas->Pen->Color = (Graphics::TColor) random(65535);
 Canvas->Pen->Width = random(7);
 Canvas->RoundRect(x, y, x + random(100), y + random(100),x, y);
}
```

4. Zadeklarujmy następujące **pliki nagłówkowe**:

```
#include <stdlib.h>
#include <time.h>
```

Ćwiczenie 8. Canvas – wykorzystanie poleceń „MoveTo” oraz „LineTo” do rysowania (zdarzenia OnMouseDown i OnMouseMove)

1. W zdarzenie dla formularza **„OnMouseMove”** wstawmy kod:

```
Canvas->MoveTo (100, 0);
Canvas->LineTo (X, Y);
```

2. W zdarzenie dla formularza **“OnMouseDown”** wstawmy kod:

```
Canvas->Pen->Color =clBlue;
Canvas->MoveTo (0,100);
Canvas->LineTo (X, Y);
```

Ćwiczenie 9. Tworzenie bitmapy z wyświetlanego formularza

1. Na formularz wprowadźmy przycisk **Button**, w jego zdarzenie wprowadźmy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
Graphics::TBitmap *FormBitmap = new Graphics::TBitmap();
FormBitmap->Handle=CreateCompatibleBitmap
(GetWindowDC(Handle), Width,  Height);
BitBlt(FormBitmap->Canvas->Handle,  0,  0,  FormBitmap->Width,
FormBitmap->Height,  GetWindowDC(Handle), 0, 0, SRCCOPY);
FormBitmap->SaveToFile("c:\\bitmapa.bmp");
delete FormBitmap;
}
```

Ćwiczenie 10. Gradientowe tło Formularza

1. W zdarzeniu **OnPaint** dla formularza umieśćmy kod:

```
void __fastcall TForm1::FormPaint(TObject *Sender)
{
  int Height;
  Height = (ClientHeight + 255) / 256 ;
  for (int Row = 0; Row <= 255; Row++)
  {
    Canvas->Brush->Color = RGB(0, 0, 255-Row);
    Canvas->FillRect(Rect(0, Row * Height, ClientWidth, (Row + 1) *
Height)) ;
  }
}
```

2. W zdarzeniu **OnResize** dla formularza umieśćmy kod:

```
void __fastcall TForm1::FormResize(TObject *Sender)
{
  Invalidate();//po zmianie rozmiarów formularza
                //tło zostanie odnowione
}
```

Ćwiczenie 11. Zmiana postaci kursora

Zmiany można dokonać ustawiając odpowiednią wartość we właściwości **Cursor** dla danego obiektu lub przez wpisanie odpowiedniego kodu.

1. Dla zdarzenia formularza **OnCreate** wprowadźmy kod:

```
TCursor orginal=Cursor;   //zapamiętanie orginalnych ustawień kursora
Cursor = TCursor (crDrag);        // wprowadzenie zmian
```

2. Dla zdarzenia **OnClose** formularza wprowadźmy kod przywracający po jego zamknięciu oryginalne ustawienia kursora:

```
TCursor orginal=Cursor;
Cursor = orginal;
```

Ćwiczenie 12. komponent CheckBox - tworzenie i sterowanie stronami TPageControl oraz polem Edit

1. Na formularzu umieszczamy komponent **PageControl** i w jego zdarzenie **OnChange** wprowadzamy kod:

```
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}
```

2. Na formularz wprowadzamy komponent **CheckBox** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}
```

3. W zdarzenie formularza **OnCreate** wprowadzamy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this);
    pPage->PageControl = PageControl1;

    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this);
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Ćwiczenie 13. Pole ComboBox i podgląd w polu RichEdit, przeglądanie zainstalowanych czcionek

1. Na formularzu umieśćmy komponenty ComboBox i RichEdit.
2. We właściwości **Font** / **Size** pola **RichEdit** zmieńmy rozmiar czcionki np. na 14 pt. a we właściwości **Lines** wprowadźmy przez edytor łańcuch np. „Przykładowy tekst."
3. We właściwości **Text** pola **ComboBox** wykasujmy standardowy tekst i wpiszmy np. „Wybierz czcionkę !!!".
4. W zdarzeniu formularza **OnCreate** umieśćmy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
    ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
}
```

5. W zdarzeniu **OnClick** pola **ComboBox** wpiszmymy kod:

```
void __fastcall TForm1::ComboBox1Click(TObject *Sender)
 {
        RichEdit1->Font->Name=ComboBox1->Items->Strings[ComboBox1-
>ItemIndex];
  }
```

Ćwiczenie 14. Dostęp do bazy DEMO (animals.dbf)

1. Umieszczamy na formularzu komponent **Table**, we właściwości **DatabaseName** wybieramy sterownik BCDEMOS, w **TableName** wybierzmy animals.dbf, ustawiamy właściwość **Active** na true.
2. Wstawmy na formularz komponent **DataSource**, jego właściwość **DataSet** ustawiamy na Table1.
3. Kolejnym elementem na formularzu jest **DBNavigator**, jego właściwość **DataSource** zmieńmy na DataSource1.
4. Ostatnim komponentem na formularzu jest **DBImage**, jego właściwość **DataSource** zmieńmy na DataSource1 a **DataField** ustawiamy na BMP.

Ćwiczenie 15. Dynamiczne tworzenie DirectoryListBox, Splitter, FileListBox

1. Do zdarzenia formularza **„OnCreate”** wprowadzamy kod:

```
{
 TDirectoryListBox *Dirs = new TdirectoryListBox (this);
 Dirs->Parent = this;
 Dirs->Align = alLeft;
 TSplitter *Split = new TSplitter(this);
 Split->Parent = this;
 Split->Left = Dirs->Left + Dirs->Width + 1;
 Split->Align = Dirs->Align;
 Split->MinSize = Form1->ClientWidth / 4;
 TFileListBox *Files = new TFileListBox(this);
 Files->Parent = this;
 Files->Align = alClient;
 Dirs->FileList = Files;
}
```

2. Do pliku dołączamy następujące wywołania :

```
#include <extctrls.hpp>
#include <filectrl.hpp>
```

Ćwiczenie 16. Program operujący na schowku (wykorzystanie komponentu Edit)

Z karty Standard umieśćmy na formularzu trzy przyciski **Button** oraz pole **Edit.**

1. Pierwszy przycisk – kopiowanie zawartości pola **Edit** do schowka**.** Parametr **Caption** przycisku na kopiuj. **OnClick** na:
   Edit1->CopyToClipboard();
2. Drugi przycisk: **Caption** na Wklej, a **onClick** na
   Edit1->PasteFromClipboard();
3. Trzeci przycisk **Caption** na Kasowanie a **onClick** na
   Edit1->Clear();
4. We właściwość **Hint** poszczególnych przycisków oraz pola **Memo** wpiszmy podpowiedzi oraz ustawiamy własność **ShowHint** na true.
5. We własności **Text** komponentu **Edit** wykasujmy znajdujący się tam łańcuch.

Ćwiczenie 17. Zliczanie ilości komponentów umieszczonych na formularzu w polu Edit

Na formularzu umieść komponent **Button** i **Edit**, w zdarzenie **OnClick** dla przycisku wpisujemy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  AnsiString nameString("TButton");
```

```
            char compBuf[10];
            TButton * button;
            for(int i=0; i < ComponentCount; i++)
            {
                    if (Components[i]->ClassNameIs(nameString))
                    {
                            button = (TButton *)Components[i];
                            button->Font->Name = "Courier";
                            itoa(ComponentCount, compBuf, 10);
                            Edit1->Text = AnsiString(compBuf) + AnsiString("
components");
                    }
        }
        }
```

1. Zapisujemy pliki i kompilujemy program.
2. Wprowadźmy na formularz dodatkowy komponent np. **Label** i ponownie kompilujemy program.

Ćwiczenie 18. Funkcja ClassName - wyświetlanie nazw obiektów na formularzu w polu Edit

1. Wprowadźmy na formularz komponent **Button**, **CheckBox**, **Label** oraz **Edit**, w zdarzenie **OnClick** przycisku wprowadźmy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        Edit1->Text = String(Button1->ClassName());
}
```

2. W zdarzenie **OnClick** komponentu **CheckBox** wprowadźmymy kod:

```
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
        Edit1->Text = String(CheckBox1->ClassName());
}
```

3. W zdarzenie **OnClick** komponentu **Label** wprowadźmy kod:

```
void __fastcall TForm1::Label1Click(TObject *Sender)
{
        Edit1->Text = String(Label1->ClassName());
}
```

Ćwiczenie 19. Edycja tekstu w polu Edit - ustawianie właściwości zamiany znaków (CharCase)

1. Wprowadźmy na formularz komponent **Edit**, trzy komponenty **RadioButton**.

2. Do zdarzenia **OnClick** dla pierwszego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
 {
        Edit1->CharCase = ecLowerCase;
}
```

3. Do zdarzenia **OnClick** dla drugiego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
        Edit1->CharCase = ecUpperCase;
}
```

4. Do zdarzenia **OnClick** dla trzeciego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
        Edit1->CharCase = ecNormal;
}
```

Podobny efekt uzyskamy bez konieczności korzystania z dodatkowych przycisków, zmieniając ustawienia we właściwości **CharCase** pola **Edit**.

Ćwiczenie 20. Dynamiczne tworzenie DirectoryListBox, Splitter, FileListBox

1. Do zdarzenia formularza „**OnCreate**” wprowadzamy kod:

```
{
 TDirectoryListBox *Dirs = new TdirectoryListBox (this);
 Dirs->Parent = this;
 Dirs->Align = alLeft;
 TSplitter *Split = new TSplitter(this);
 Split->Parent = this;
 Split->Left = Dirs->Left + Dirs->Width + 1;
 Split->Align = Dirs->Align;
 Split->MinSize = Form1->ClientWidth / 4;
 TFileListBox *Files = new TFileListBox(this);
 Files->Parent = this;
 Files->Align = alClient;
 Dirs->FileList = Files;
}
```

2. Do pliku dołączamy następujące wywołania :

```
#include <extctrls.hpp>
#include <filectrl.hpp>
```

Ćwiczenie 21. Wywoływanie wielu formularzy w jednej aplikacji

1. Formularz nr 1 nazwijmy „Formularz podstawowy.” (w **Caption**).
2. Osadźmy na formularzu dwa przyciski **BitBtn** z karty **Additional.**

3. Pierwszy przycisk nazwijmy „Otwórz Formularz 2” (w **Caption**).
4. Drugi nazwijmy „Otwórz Formularz 3” (w **Caption**).
5. Przez własność **Glyph** dodajmy do przycisków obrazki.
6. Dostosujmy rozmieszczenie rys. przez własność **Layout**, kolor przez **Color**, właściwości czcionki przez **Font**, podpowiedź przez **Hint**, ustawiamy **ShowHint** na True.
7. Zapisujemy projekt na dysku np. poleceniem **File/Save All.**
8. Przez polecenie **File/New** Form tworzymy nowy formularz i nazywamy go „Formularz 2” (w **Caption**), zmniejszamy jego rozmiary aby nam nie przysłaniał innych formularzy.
9. Powyższą czynność powtarzamy tworząc „Formularz 3”.
10. Opisujemy zdarzenia dla przycisków:

Dla przycisku nr 1 w **onClick** :

```
Form2->Show();
```

Dla przycisku nr 2 w **onClick**:

```
Form3->Show();
```

Jeśli chcemy utworzyć okna tzw. modalne zmieńmy funkcje „**Show()**” na **ShowModal()**” Przed kompilacją programu należy go zapisać na dysku i dołączyć do pliku „Unit1.cpp” deklarację klas zawartych w „Unit2.cpp” oraz „Unit3.cpp” możemy to wykonać przez wybranie polecenia **File/Include Unit Hdr....**

Ćwiczenie 22. Tworzenie formularza z ramką

1. W pliku nagłówkowym **Unit1.h** wpisujemy deklarację funkcji **CreateParams** w sekcji **private** :

```
private:
void __fastcall CreateParams(TCreateParams &Params);
```

2. W pliku **Unit1.cpp** rozwijamy funkcję:

```
void __fastcall TForm1:: CreateParams (TCreateParams  &Params)
{
TForm::CreateParams(Params);
Params.ExStyle |= WS_EX_CLIENTEDGE;
```

```
}
```

Ćwiczenie 23. Przesuwanie formularza po uchwyceniu dowolnego jego punktu

1. W pliku nagłówkowym **Unit1.h** deklarujemy funkcję **MoveIt** w sekcji **public**:

```
void __fastcall MoveIt (TMessage &Msg);
BEGIN_MESSAGE_MAP
   MESSAGE_HANDLER(WM_NCHITTEST, TMessage, MoveIt)
END_MESSAGE_MAP(TForm)
```

2. W pliku **Unit1.cpp** rozwijamy funkcję:

```
void __fastcall TForm1::MoveIt(TMessage &Msg)
{
TForm :: Dispatch (&Msg);
if (Msg.Result == HTCLIENT) Msg.Result = HTCAPTION;
}
```

Ćwiczenie 24. Określanie stylu formularza

Wprowadźmy na formularz pole „Edit". Do pierwszej funkcji kodu źródłowego wprowadźmy następujący kod:

```
__fastcall TForm1::TForm1(TComponent* Owner)
                                : TForm(Owner)
{
if(FormStyle != fsMDIForm)
FormStyle = fsMDIForm;
if(FormStyle == fsMDIForm)
Edit1->Text = "MDI form";
else
Edit1->Text = "Not an MDI form";
}
```

Ćwiczenie 25. Wywoływanie zminimalizowanego formularza

1. Wprowadźmy na formularz 2 przyciski „**Button**", do pierwszego przycisku zainicjujmy następujące zdarzenie na kliknięcie:
   ShowWindow(Form2->Handle, SW_SHOWMINIMIZED);
2. Do drugiego przycisku zainicjujmy zdarzenie:
   ShowWindow(Form2->Handle, SW_SHOWMINNOACTIVE);
3. Utwórzmy drugi formularz (**Form2**).
4. Do formularza Form1 dołączmy wywołanie „**Unit2.h**" – przez **File/Include Unit HDR.... .**

Ćwiczenie 26. Podświetlanie aktywowanego formularza przez zmianę jego koloru (przy pracy z kilkoma formularzami jednocześnie)

1. Na formularz wprowadzamy przycisk Button, w jego zdarzenie OnClick wpisujemy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Form2->Show();
}
```

2. W pliku **Unit1.h** sekcji **public** deklarujemy metodę **ColorForm**:

```
void __fastcall TForm1::ColorForm(TObject *Sender)
```

3. W pliku **Unit1.cpp** rozpisujemy metodę **ColorForm**:

```
void __fastcall TForm1::ColorForm(TObject *Sender)
{
  Color = clBtnFace;
  Form2->Color = clBtnFace;
  Screen->ActiveForm->Color = clAqua;
}
```

4. W zdarzeniu formularza **OnCreate** wpisujemy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  Screen->OnActiveFormChange = ColorForm;
}
```

5. Poleceniem **File/New Form** tworzymy kolejny formularz (**Form2**).
6. Do **Form1** dołączamy plik **Unit2** poleceniem **File/Include Unit Hdr** ... lub ALT+F11.

Ćwiczenie 27. Komunikat – reakcja na naciśnięcie dowolnego klawisza z jego wyszczególnieniem

Do zdarzenia formularza **OnKeyPress** dopisujemy kod:

```
void __fastcall TForm1::FormKeyPress(TObject *Sender, char &Key)
{
  char keyString[25];
  keyString[0] = Key;
  strcpy(&keyString[1], "- nacisnąłeś klawisz");
  Application->MessageBox(keyString , "Key Press", MB_OK);
}
```

Ćwiczenie 28. Wyświetlanie pozycji wskaźnika myszy na formularzu

1. Zadeklarujmy plik nagłówkowy :
   #include <stdlib.h> //dotyczy funkcji itoa
2. W lewym górnym rogu formularza umieśćmy dwie etykiety **Label**.
3. W nazwę (**Name**) pierwszej wpiszmymy „Poziom”, a drugiej „Pion”.
4. Do zdarzenia formularza **OnMouseMove** wpiszmymy kod:

```
void __fastcall TForm1::FormMouseMove(TObject *Sender, TShiftState
Shift,
    int X, int Y)
{
        char xPos[10];
        char yPos[10];
        itoa(X, xPos, 10);
        itoa(Y, yPos, 10);
        Poziom->Caption = xPos;
        Pion->Caption = yPos;
}
```

Ćwiczenie 29. Zamykanie aplikacji, przez wywołanie okienka dialogowego

1. Do zdarzenia **OnCloseQuery** Formularza wpisujemy kod:
   if(Application->MessageBox("Close the form?",
   "Close?",MB_YESNOCANCEL) != mrYes) CanClose = false;

Ćwiczenie 30. Wyświetlanie bieżącej daty i czasu przy aktywacji formularza

1. Wprowadźmy na formularz dwa komponenty **Label**.
2. Zmieńmy ich właściwości **Font** / **Size** np. na 12 pt.
3. W zdarzenie formularza **OnCreate** wprowadźmy kod:

```
TDateTime data_biezaca = TDateTime::CurrentDate();
Label1->Caption = data_biezaca.FormatString("dd-mm-yyyy");
TDateTime czas_biezacy = TDateTime::CurrentTime();
Label3->Caption = czas_biezacy.FormatString("hh-nn-ss");
```

Ćwiczenie 31. Tworzenia formularza bez obramowania

1. Ustawiamy wszystkie podwłaściwości **BorderIcons** na false.
2. Ustawiamy **BorderStyle = bsNone**.
3. Wyczyścić zawartość **Caption**.

Ćwiczenie 32. Otwieranie formularza przy dźwiękach plików typu *.wav

1. W zdarzenie formularza **OnCreate** wprowadzamy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  sndPlaySound("chord.wav", SND_SYNC);
  Close();
}
```

2. Jeśli plik dźwiękowy znajduje się poza katalogiem gromadzącym pliki aplikacji to należy podać do niego pełną ścieżkę dostępu np.

„ c:\\ muzyka\\ muz.wav”

3. Deklarujemy również plik nagłówkowy:

```
#include <mmsystem.h>
```

Ćwiczenie 33. Usunięcie ikon minimalizacji, maksymalizacji i zamknięcia formularza

Wprowadźmy na formularz przycisk **Button**, w jego zdarzenie **OnClick** wpiszmy kod:

```
BorderIcons = BorderIcons - (TBorderIcons()<< biMaximize);
```

Ćwiczenie 34. Wyświetlenie mapy bitowej w oknie formularza

1. Zmieńmy nagłówek formularza **Caption** na „Wyświetlanie obrazu”.
2. Zakładka Additional komponent **Image** –Wstawmy do formularza.
3. Właściwość **Align** (wyrównanie) komponentu **Image** na **alClient** (wypełnienie całej powierzchni formularza).
4. Właściwość **Stretch** (skalowanie) na true (wraz ze zmianą wielkości formularza zmieniana będzie wartość obrazka).
5. Właściwość **Picture** (treść obrazka) –wstawianie obrazka np. z /ProgramFiles/ Common Files/ Borland Shared/ Images/ Splash/256Color/.....

Ćwiczenie 35. Wykorzystanie Image w programie tworzącym przeglądarkę plików BMP

1. Wybieramy komponent **ScrollBox**, do jego wnętrza dodajemy komponent **Image.**
2. Z karty **Dialogs** pobieramy **OpenDialog** i ustawiamy własność **Filter** na **Pliki BMP** oraz *.bmp
3. Ustawiamy wartość **AutoSize** obiektu **Image** na true.
4. Wstawiamy przycisk do pobierania obrazków i w jego własność **onClick** wpisujemy :

```
if (OpenDialog1->Execute())
Image1->Picture->LoadFromFile(OpenDialog1->FileName);
```

Ćwiczenie 36. Image – czyszczenie pola

1. Wprowadzamy na formularz komponent **Image** oraz **Button**, w zdarzenie **OnClick** dla Button wpisujemy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Image1->Picture = NULL;
}
```

2. Dla komponentu **Image** przez właściwość **Picture** wprowadzamy wybrany obrazek (*.bmp). Wcześniej pole **Image** dostosowujemy do naszych potrzeb (np. przez właściwości Align, Height, Width, Transparent itp.).

Ćwiczenie 37. Efekt obrotu bitmapy o 90 stopni

1. Wprowadzamy na formularz pole **Image** i wybieramy dla niego obraz przez właściwość **Picture**.
2. Na formularz wprowadzamy przycisk **Button** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int x, y;
  int width, height;
  Graphics::TBitmap* tmpBMP = new Graphics::TBitmap;

  tmpBMP->Height = Image1->Width;
  tmpBMP->Width = Image1->Height;
  width = Image1->Width-1;
  height = Image1->Height-1;

  for (y = 0; y <= height; y++)
    for (x = 0; x <= width; x++)
    {
        tmpBMP->Canvas->Pixels[y][width-x]=Image1->Canvas->Pixels[x]
[y];
       Application->ProcessMessages();
    }
   Image1->Width = tmpBMP->Width;
  Image1->Height = tmpBMP->Height;
  Image1->Picture->Bitmap->Assign(tmpBMP);
  delete tmpBMP;
}
```

Ćwiczenie 38. Tworzenie negatywu obrazu

1. Wprowadzamy na formularz pole **Image** i wybieramy dla niego obraz przez właściwość **Picture**.
2. Na formularz wprowadzamy przycisk **Button** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  for (int i=0; i<=Image1->Height-1; i++)
    for (int j=0; j<=Image1->Width-1; j++)
    {
      TColor Kolor = Image1->Picture->Bitmap->Canvas->Pixels[j][i];
      byte r,g,b;
      r = 255 - GetRValue(Kolor);
      g = 255 - GetGValue(Kolor);
      b = 255 - GetBValue(Kolor);
      Image1->Picture->Bitmap->Canvas->Pixels[j][i] = RGB(r,g,b);
    }
}
```

Ćwiczenie 39. Efekt przygaszania bitmapy

1. Do kodu dołączamy plik nagłówkowy : **#include <stdlib.h>**
2. Na formularzu umieszczamy pole **Image** oraz przycisk **Button** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  int BytesPerScan = abs(int(Image1->Picture->Bitmap->ScanLine[1]) -
                         int(Image1->Picture->Bitmap->ScanLine[0]));
  for (int i = 1; i <= 256; i++)
  {
    for (int lineNr = 0; lineNr <= Image1->Picture->Bitmap->Height-1; lineNr++)
    {
      Byte* p = (Byte*)Image1->Picture->Bitmap->ScanLine[lineNr];
      for (int x = 0; x <= BytesPerScan - 1; x++)
        if (p[x] > 0) p[x] = --p[x];
      Application->ProcessMessages();
    }
    Image1->Refresh();
  }
}
```

Ćwiczenie 40. Tworzenie bannera

1. Przygotujmy sobie kilka obrazów w formacie ***.bmp** mających tworzyć baner.
2. Ustawmy odpowiednio do wielkości bannera właściwości **Height** i **Width** komponentu **ImageList**.
3. Wczytajmy do **ImageList** poszczególne klatki animacji bannera (należy ustawić **Transparent Color** na **clNone** po wybraniu obrazków - dla każdego z nich).
4. Umieszczamy na formularzu komponent **Timer**.
5. Ustawiamy właściwość **Interval** komponentu Timer np.na 150 ms.
6. Wstawmy na formularz komponent **Image**, na którym wyświetlany będzie banner, ustawiamy jego właściwości **Height** i **Width**.
7. W pliku nagłówkowym formularza w sekcji **private** tworzymy zmienną:

```
private:
  int nrObrazu;
```

8. Wstawiamy do komponentu **Image1** obrazek z pierwszą klatką bannera.
9. Tworzymy zdarzenie **OnTimer** dla Timera:

```
void __fastcall TForm1::TimerTimer(TObject *Sender)
{
  Image1->Canvas->FillRect(Rect(0, 0, Image1->Width, Image1->Height));
  ImageList1->Draw(Image1->Canvas, 0, 0, nrObrazu);
  if (++nrObrazu > ImageList1->Count - 1) nrObrazu = 0;
}
```

10. Ustawmy właściwość **Cursor** komponentu **Image1** na **crHandPoint**.
11. Utwórzmy funkcję obsługującą zdarzenie **OnClick** dla komponentu **Image**:

```
void __fastcall TForm1::Image1Click(TObject *Sender)
{
  Image1->Canvas->FillRect(Rect(0, 0, Image1->Width, Image1->Height));
}
```

Ćwiczenie 41. Przyciski z podwójnymi ikonami w ToolBar (wykorzystanie komponentu ImageList)

1. Na formularzu umieszczamy **ToolBar.**
2. Klikamy prawym klawiszem myszy w jego obszarze i wybieramy **New Button**, możemy utworzyć także odstępy między przyciskami wybierając **New Separator**. Tworzymy na nim: Button, Separator, Button, Separator, Button. Nazywamy (**Name**) kolejne utworzone przyciski Jeden, Dwa, Trzy.
3. Zmieniamy właściwości **ToolBar**:
   EdgeBorders
   ebBottom true //wyświetlanie dolnej krawędzi
   Flat true //przyciski bez obramowania
4. Na formularzu umieszczamy 2 komponenty **ImageList**, **ImageList1** będzie zawierał obrazki wyświetlane na pasku, a **ImageList2** obrazki wyświetlane na pasku po najechaniu na niego myszą.
5. Klikamy na **ImageList1** i za pomocą przycisku **Add**... dodajemy po kolei trzy dowolne ikony (ProgramFiles\CommonFiles\BorlandShared\Images\Icons).
6. Tak samo postępujemy z **ImageList2** ale wybieramy inne ikony (ikony można sobie przygotować wcześniej w dowolnym programie graficznym).
7. We właściwości **Images** (dla **ToolBar**) ustawiamy **ImageList1** a dla właściwości **HotImages** ustawiamy **ImageList2**.

Ćwiczenie 42. Blokowanie kombinacji ALT+TAB

1. Wprowadźmy na formularz przycisk **Button**, do zdarzenia na kliknięcie **OnClick** wprowadźmy następujący kod (blokujący działanie ALT-TAB):

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  LongBool OldValue;
SystemParametersInfo(97, Word(true), &OldValue, 0);
  }
```

2. Wprowadźmy na formularz kolejny przycisk **Button**, do zdarzenia na kliknięcie **OnClick** wprowadźmy następujący kod (odblokowujący działanie ALT-TAB):

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 LongBool OldValue;
  SystemParametersInfo(97, Word(false), &OldValue, 0);
 }
```

Ćwiczenie 43. Migające podświetlenie etykiety Label

1. Wprowadźmy na formularz komponent **Timer** oraz **Label**.
2. Właściwość **ParentColor** komponentu **Label** ustawiamy na **true**.
3. Do zdarzenia **OnTimer** komponentu **Timer** wprowadźmy kod:

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
        if(Label1->ParentColor)
                Label1->Color = clRed;
        else
                Label1->ParentColor = true;
}
```

Ćwiczenie 44. Komponent ListBox – wstawianie elementów, wyróżnianie elementów podobnych przy wpisywaniu na listę

1. Do formularza wstawiamy ListBox.
1. Wstawiamy też pole edycji **Edit** – posłuży do wstawiania kolejnego elementu listy.
2. Nad polem edycji umieszczamy etykietę z napisem „Kolejny element listy".
3. Wstawmy przycisk z napisem w **Caption** „Dodaj do listy", w jego własność **OnClick** wpiszmy kod:

```
ListBox1->Items->Add(Edit1->Text);
```

4. Wpisujemy we własność **OnClick** komponentu ListBox :

```
Edit1->Text=ListBox1->Items->Strings[ListBox1->ItemIndex];
```

5. Wykasujmy we właściwości **Text** pola Edit znajdujący się w nim zapis.
6. Wstawmy następny przycisk nazwijmy go „Kasuj elementy" i we właściwość **OnClick** wstawmy :

```
ListBox1->Items->Delete(0);
```

Wyróżnianie elementu podobnego do wpisywanego, przez komunikat **LB_FINDSTRING** , wpisujemy poniższy kod do zdarzenia pola edycji **onKeyUp** :

```
int index = ListBox1->Perform (LB_FINDSTRING, -1,
        (LPARAM) Edit1->Text.c_str());
if (index != LB_ERR) ListBox1->ItemIndex = index;
```

7. Kompilujemy program, zapisując go wcześniej na dysku.

Ćwiczenie 45. Odczytywanie zawartości pliku tekstowego za pomocą metody LoadFromFile()

1. Wstawmy do formularza komponent **ListBox**.
2. Opisujemy funkcję **FormCreate()** dla formularza, uzupełniając jej kod:

```
char winDir[256], fileName[256];
GetWindowsDirectory (winDir, sizeof (winDir));
sprintf (fileName, "%s\\win.ini", winDir);
ListBox->Items->LoadFromFile(fileName);
```

Ćwiczenie 46. Tworzenie listy komponentów w polu ListBox

1. Wprowadźmy na formularz komponent **ListBox** oraz kilka innych np. **Button**, **StatusBar**, **StaticText** itp.
2. Dla komponentu **StatusBar** ustawiamy właściwość **SimplePanel** na **true**.
3. Wpiszmy w zdarzenie **OnCreate** dla formularza kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
 {
  for (int i = 0; i < ComponentCount; i++)
   ListBox1->Items->InsertObject(0,
                          Components[i]->Name,
                          (TObject *)Components[i]);

}
```

4. W zdarzenie **OnMouseUp** dla **ListBox** wpiszmy kod:

```
void __fastcall TForm1::ListBox1MouseUp(TObject *Sender,
TMouseButton Button, TShiftState Shift, int X, int Y)
{
 if (Button == mbRight)
 {
  TClass ClassRef;
  int Index = ListBox1->ItemAtPos(Point(X,Y), true);
  // only components that are controls have a position
  // make sure the component is a control
  for (ClassRef = ListBox1->Items->Objects[Index]->ClassType();
      ClassRef != NULL;
      ClassRef = ClassRef->ClassParent())
   if (String(ClassRef->ClassName()) == "TControl")
   {
    TControl *TheObject = (TControl *)ListBox1->Items-
>Objects[Index];

     StatusBar1->SimpleText =
      TheObject->Name + " is at (" +
      IntToStr(TheObject->Left) + ", " +
      IntToStr(TheObject->Top) + ")";
     break;
   }
  if (ClassRef == NULL) // if it wasn't a control
   MessageBeep(0);
 }
}
```

Ćwiczenie 47. Program demonstrujący możliwości MaskEdit

1. Wstawiamy etykietę i dajemy jej dowolną nazwę np.:
   „Numer Telefonu”
2. Obok etykiety wstawiamy obiekt **MaskEdit** z karty Additional, we własność **EditMask** wpisujemy kod przez edytor:
   !\(\099\)00-00-00;1;_ (możemy wybrać z edytora)

Ćwiczenie 48. Wykorzystanie komponentu MediaPlayer

1. Wstawiamy na formularz komponent **MediaPlayer** oraz np. przycisk **BitBtn**.
2. Do zdarzenia OnClick przycisku wstawiamy kod:
   MediaPlayer1->AutoEnable = false;
   MediaPlayer1->EnabledButtons.Clear();

Ćwiczenie 49. Dodanie Menu do formularza

1. Z karty Standard wybieramy komponent **MainMenu** i wstawiamy do formularza.
2. Przez własność **Items** a następnie własność **Captions** wprowadzamy pierwszą nazwę:
   Kolor
   Przez klawisze Enter, Insert i Delete modyfikujemy wygląd Menu
   Wprowadzając kolejno
   Zielony
   Czerwony
   Niebieski
3. Opisujemy reakcje na kliknięcie poszczególnych kolorów **Menu**:
   Z Karty **Evens** wybieramy **OnClick** dla koloru Zielony i wpisujemy:
   Form1->Color=clGreen;
   Dla pozostałych kolorów postępujemy podobnie
4. Do **Caption** formularza wprowadzamy nazwę:
   Zmiana kolorów formularza przez Menu.

Ćwiczenie 50. Menu podręczne (wykorzystanie komponentu PopupMenu)

1. W formularzu umieszczamy **Panel**.
2. Na formularzu umieszczamy dwa komponenty **PopupMenu.**
3. W **PopupMenu1** dodajemy pozycje zielony i czerwony (będą one zmieniały kolor Panelu), dopisujemy w OnClick dla każdego koloru:
   Panel1->Color=clGreen; (dla czerwony podobnie).
4. **PopupMenu2** dodajemy pozycję niebieski i szary będą zmieniały kolor Formularza,I dopisujemy w OnClick dla każdego koloru:
   Form1->Color=clBlue; (dla szarego podobnie).
5. Podpinamy **PopupMenu1** pod własność **PopupMenu** dla **Panel** oraz **PopupMenu2** pod własność **PopMenu** dla Formularza.
6. Po kompilacji klikając prawym przyciskiem myszy na panel możemy zmieniać jego kolor, to samo dla formularza.

Ćwiczenie 51. Osadzanie w formularzu arkusza Excela – wykorzystanie komponentu OleContainer

1. Z karty System wybieramy **OleContainer**, z karty Standard przycisk **Button** oraz **ListBox.**
2. Z **OleContainer** przez prawy klawisz myszy wybieramy „**InsertObject**" a następnie „Arkusz Microsoft Excel".
3. Przycisk nazywamy Pobieranie i w **onClick** wpisujemy:

```
ListBox1->Items=OleContainer1->ObjectVerbs;
```

4. W ListBox **onClick** wstawiamy:

```
OleContainer1->DoVerb(ListBox1->ItemIndex);
```

Ćwiczenie 52. Tworzenie i sterowanie stronami komponentu TPageControl.

1. Wybieramy komponent **PageControl** i wstawiamy go na formularz.
2. W zdarzenie **OnCreate** Formularza wpisujemy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
 for (int i = 0; i < 10; i++)
 {
  TTabSheet *pPage = new TTabSheet(PageControl1);
  pPage->PageControl = PageControl1;
  pPage->Caption = AnsiString("Strona") + IntToStr(i);
 }
}
```

3. Na formularz wprowadzamy komponent **UpDown** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::UpDown1Click(TObject *Sender, TUDBtnType
Button)
{
 PageControl1->SelectNextPage(Button == btNext);
}
```

4. Ustawiamy właściwości **UpDown** :

- enable na false
- visible na false

Ćwiczenie 53. Komponent CheckBox - tworzenie i sterowanie stronami TPageControl oraz polem Edit

1. Na formularzu umieszczamy komponent **PageControl** i w jego zdarzenie **OnChange** wprowadzamy kod:

```
void __fastcall TForm1::PageControl1Change(TObject *Sender)
{
  CheckBox1->Checked = PageControl1->ActivePage->Visible;
}
```

2. Na formularz wprowadzamy komponent **CheckBox** a w jego zdarzenie **OnClick** wprowadzamy kod:

```
void __fastcall TForm1::CheckBox1Click(TObject *Sender)
{
  PageControl1->ActivePage->Visible = CheckBox1->Checked;
}
```

3. W zdarzenie formularza **OnCreate** wprowadzamy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < 10; i++)
  {
    TTabSheet *pPage = new TTabSheet(this);
    pPage->PageControl = PageControl1;

    pPage->Caption = AnsiString("Page") + IntToStr(i);
    TEdit *pEdit = new TEdit(this);
    pEdit->Parent = pPage;
    pEdit->Left = random(pPage->ClientWidth - pEdit->Width);
    pEdit->Top = random(pPage->ClientHeight - pEdit->Height);
  }
  PageControl1Change(Sender);
}
```

Ćwiczenie 54. Zastosowanie w komponentu typu Panel formularzu

1. Z karty standard komponent **Panel** Wstawmy do formularza.
2. Z właściwości **Align** pobierajmy po kolei poszczególne elementy, zaobserwujmy jak będzie rozmieszczany panel na obszarze formularza i czy można go w danej postaci skalować, zmieniać rozmiar czy też nie.

Ćwiczenie 55. Dynamiczne tworzenie ProgressBar

Na formularzu umieśćmy przycisk „**Button**”, w zdarzeniu na kliknięcie umieśćmy następujący kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  TProgressBar *ProgressBar = new TProgressBar(this);
  ProgressBar->Parent = this;
  ProgressBar->Align = alBottom;
}
```

Ćwiczenie 56. Edycja tekstu w polu Edit ustawianie właściwości zamiany znaków (CharCase) - małe, wielkie litery

1. Wprowadźmy na formularz komponent **Edit**, trzy komponenty **RadioButton**.
2. Do zdarzenia **OnClick** dla pierwszego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton1Click(TObject *Sender)

{
        Edit1->CharCase = ecLowerCase;
}
```

3. Do zdarzenia **OnClick** dla drugiego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
        Edit1->CharCase = ecUpperCase;
}
```

4. Do zdarzenia **OnClick** dla trzeciego **RadioButton** wpiszmy kod:

```
void __fastcall TForm1::RadioButton3Click(TObject *Sender)
{
        Edit1->CharCase = ecNormal;
}
```

Podobny efekt uzyskamy bez konieczności korzystania z dodatkowych przycisków, zmieniając ustawienia we właściwości **CharCase** pola Edit.

Ćwiczenie 57. Przeglądanie zainstalowanych czcionek przez wybór z pola ComboBox i podgląd w polu RichEdit

1. Na formularzu umieśćmy komponenty **ComboBox** i **RichEdit**.
2. We właściwości **Font** / **Size** pola **RichEdit** zmieńmy rozmiar czcionki np. na 14 pt. a we właściwości **Lines** wprowadźmy przez edytor łańcuch np. „Przykładowy tekst."
3. We właściwości **Text** pola **ComboBox** wykasuj standardowy tekst i wpiszmy np. „Wybierzmy czcionkę !!!".
4. W zdarzeniu formularza **OnCreate** umieśćmy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
  for (int i = 0; i < Screen->Fonts->Count; i++)
    ComboBox1->Items->Add(Screen->Fonts->Strings[i]);
}
```

5. W zdarzeniu **OnClick** pola **ComboBox** wpiszmy kod:

```
 void __fastcall TForm1::ComboBox1Click(TObject *Sender)
  {
            RichEdit1->Font->Name=ComboBox1->Items->Strings[ComboBox1-
>ItemIndex];
   }
```

Ćwiczenie 58. Odczyt, zapis plików tekstowych - szkielet aplikacji

1. Właściwość **Name** formularza ustawiamy na Cwiczenie1.
2. Na formularz wprowadzamy pole **Memo**, nadajemy mu nazwę (**Name**) Memo, ustalamy rozmiar dowolnie rozciągając.
3. Z karty Standard wybieramy **Panel** ustawiamy **Name** na Podkladka, **Caption** wyczyść, **BevelOuter** na bvNone, **Align** na alTop.
4. Z karty Additional wybierzmy **SpeedButton** i Wstawmy do panelu, **Name** na Open, **Left** na 5.
5. Przez właściwość **Glyph** – z Program Files/Common Files/ Borland Shared/ Images/ Buttons wstawmy ikonkę na przycisk.
6. Wstawmy kolejne trzy przyciski do panelu. Powtórzmy wymienione czynności dla nich, pierwszy **Name** na Save, drugi **Name** na SaveAs, trzeci **Name** na Koniec.
7. Wstawmy do formularza z karty Dialog **OpenDialog** i **SaveDialog**, zmieńmy im właściwość **Name** pozostawiając nazwę i kasując nr.
8. Wybierzmy przycisk **Koniec** i dwukrotnie klikając w otwartą funkcję wpiszmy Close();
9. Dla przycisku **Open** wpiszmy kod (dwa razy klikając w niego):

```
if(Memo->Modified)
{
int result = Application->MessageBox(
"Czy zachować bieżący dokument?",
"Uwaga",MB_YESNOCANCEL | MB_ICONWARNING);
if (result==IDYES) SaveClick(0);
if (result ==IDCANCEL) return;
}
OpenDialog->FileName ="";
if (OpenDialog->Execute())
{
if (Memo->Lines->Count > 0)  Memo -> Clear();
Memo->Lines->LoadFromFile (OpenDialog->FileName);
SaveDialog->FileName = OpenDialog ->FileName;
}
```

10. Dla przycisku **Save** wprowadźmy kod klikając w niego dwa razy:

```
if (SaveDialog -> FileName !="")
{
Memo->Lines->SaveToFile (SaveDialog->FileName);
Memo->Modified =false;
}
else SaveAsClick(Sender);
```

11. Dla przycisku **SaveAs** zapisz kod klikając w niego dwa razy:

```
SaveDialog ->Title = "Zapisz jako";
if (SaveDialog -> Execute())
{
Memo->Lines->SaveToFile (SaveDialog->FileName);
Memo->Modified =false;
}
```

Ćwiczenie 59. Zastosowanie w formularzu komponentów typu Shape – ustalanie wzajemnego ich położenia

1. Z karty Additional wybierzmy komponent **Shape** (kształt) i wstawmy go do formularza.
2. Zmieńmy właściwość Shape na **stEllipse Width** na 70 a **Hight** na 170.
3. Kliknij w szare pole **Brush** i zmieńmy składową **Color** na **clBlue**.
4. Wstawmy do formularza 2 komponent **Shape**.
5. Podobnie jak w poprzednim, zmieńmy **Shape** tym razem na **stCircle** a w polu **Brush** wybierzmy **Color stYellow**.
6. Umieśćmy małe koło na elipsie.
7. Wybierzmy kliknięciem elipsę a następnie dodajmy do grupy białe koło przy wciśniętym klawiszu Shift.
8. Z Menu wybierzmy polecenia **View** i **Aligment Palette**.
9. Sprawdźmy przemieszczanie elementów za pomocą przycisków.
10. Ten sam efekt wyrównania można osiągnąć wywołując tradycyjne okno z właściwościami wyrównującymi przyciskając prawy przycisk myszki i wybierając **Alignment**.

Ćwiczenie 60. Wykorzystanie komponentów ScrollBox i Image w programie tworzącym przeglądarkę plików BMP

1. Pobieramy komponent **ScrollBox**, do jego wnętrza dodajemy komponent **Image**.
2. Z karty Dialogs pobieramy **OpenDialog** i ustawiamy własność **Filter** na Pliki BMP oraz *.bmp
3. Ustawiamy wartość **AutoSize** obiektu Image na true.
4. Wstawiamy przycisk do pobierania obrazków i w jego własność **onClick** wpisujemy :

```
if (OpenDialog1->Execute())
Image1->Picture->LoadFromFile(OpenDialog1->FileName);
```

Ćwiczenie 61. Wywołanie okna regulacji głośności

Na formularzu umieszczamy przycisk **Button** i w jego zdarzenie **OnClick** wpisujemy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  WinExec("SNDVOL32.EXE", SW_SHOWNORMAL);
}
```

Ćwiczenie 62. Odczyt bieżącego czasu

1. Na formularz wprowadzamy przycisk **Button**, etykietę **Label**.
2. W zdarzenie **OnClick** przycisku wprowadzamy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
  Label1->Caption = "Bieżąca data :" + TimeToStr(Time());
}
```

Ćwiczenie 63. Migające podświetlenie etykiety Label

1. Wprowadźmy na formularz komponent **Timer** oraz **Label**.
2. Właściwość **ParentColor** komponentu **Label** ustawiamy na **true**.
3. Do zdarzenia **OnTimer** komponentu **Timer** wprowadźmy kod:

```
void __fastcall TForm1::Timer1Timer(TObject *Sender)
{
        if(Label1->ParentColor)
                Label1->Color = clRed;
        else
                Label1->ParentColor = true;
}
```

Ćwiczenie 64. Wykorzystanie ToolBar oraz StringList do tworzenia dynamicznych przycisków

1. W górnym sektorze formularza umieśćmy komponent **ToolBar**.
2. Do pliku źródłowego dopiszmy wywołanie :

```
#include <comctrls.hpp>
```

3. W kodzie źródłowym umieśćmy następującą funkcję:

```
void AddButtons(TToolBar *pToolBar, TStringList *pCaptions)
{
for (int i = 0; i < pCaptions->Count; i++)
{
TToolButton *pButton = new TToolButton(pToolBar);
pButton->Parent = pToolBar;
pButton->Caption = pCaptions->Strings[i];
if (pButton->Caption == "|")
pButton->Style = tbsSeparator;
else
pButton->Style = tbsButton;
}
}
```

4. Dla zdarzenia "**OnCreate**" formularza dopiszmy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
TToolBar *pToolBar = new TToolBar(this);
pToolBar->Parent = this;
TStringList *pCaptions = new TStringList();
pCaptions->Add("New");
pCaptions->Add("Save");
pCaptions->Add("|");
pCaptions->Add("Cut");
pCaptions->Add("Copy");
pCaptions->Add("Paste");
AddButtons(pToolBar, pCaptions);
delete pCaptions;
pToolBar->ShowCaptions = True;
pToolBar->Height = 40;
}
```

Ćwiczenie 65. Przyciski z podwójnymi ikonami w ToolBar

1. Na formularzu umieszczamy **ToolBar.**
2. Klikamy prawym klawiszem myszy w jego obszarze i wybieramy **New Button**, możemy utworzyć także odstępy między przyciskami wybierając **New Separator**. Tworzymy na nim: Button, Separator, Button, Separator, Button. Nazywamy (**Name**) kolejne utworzone przyciski Jeden, Dwa, Trzy.
3. Zmieniamy właściwości **ToolBar**:
   EdgeBorders
   ebBottom true //wyświetlanie dolnej krawędzi
   Flat true //przyciski bez obramowania
4. Na formularzu umieszczamy 2 komponenty **ImageList**, **ImageList1** będzie zawierał obrazki wyświetlane na pasku, a **ImageList2** obrazki wyświetlane na pasku po najechaniu na niego myszą.
5. Klikamy na **ImageList1** i za pomocą przycisku **Add**... dodajemy po kolei trzy dowolne ikony (ProgramFiles\CommonFiles\BorlandShared\Images\Icons).
6. Tak samo postępujemy z **ImageList2** ale wybieramy inne ikony (ikony można sobie przygotować wcześniej w dowolnym programie graficznym).
7. We właściwości **Images** (dla **ToolBar**) ustawiamy **ImageList1** a dla właściwości **HotImages** ustawiamy **ImageList2**.

Ćwiczenie 66. Dynamiczne tworzenie listy drzewa w komponencie TreeView

Na formularzu umieszczamy komponent **TreeView** oraz **Button**, w zdarzenie przycisku **OnClick** wprowadźmy kod:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
        TreeView1->Items->Add(TreeView1->Selected, RootTreeNode1");
        TTreeNode *MyTreeNode1 = TreeView1->Items->Item[0];
        TreeView1->Items->AddChild(MyTreeNode1,"ChildNode1");
        TreeView1->Items->Add(TreeView1->Selected, RootTreeNode2");
        TTreeNode *MyTreeNode2 = TreeView1->Items->Item[2];
        TreeView1->Items->AddChild(MyTreeNode2,"ChildNode2");
        MyTreeNode2=TreeView1->Items->Item[3];
        TreeView1->Items->AddChild(MyTreeNode2,"ChildNode2a");
        MyTreeNode2 = TreeView1->Items->Item[4];
        TreeView1->Items->Add(MyTreeNode2,"ChildNode2b");
        TreeView1->Items->Add(TreeView1->Selected, RootTreeNode3");
}
```

Ćwiczenie 67. Podświetlanie pól przy wchodzeniu i wychodzeniu z nich (np. EDIT i MEMO)

1. Wprowadźmy na formularz komponent **Edit** z karty **Standard** do zdarzenia **„OnEnter”** wprowadźmy następujący kod:

```
void __fastcall TForm1::Edit1Enter(TObject *Sender)
{
Edit1->Color = clYellow;
}
```

2. Do zdarzenia **“OnExit”** wprowadźmy kolejny kod:

```
void __fastcall TForm1::Edit1Exit(TObject *Sender)
{
Edit1->Color = clWindow;
}
```

3. Na formularz wprowadźmy komponent **Memo** do zdarzenia **“OnEnter”** wprowadźmy następujący kod:

```
void __fastcall TForm1::Memo1Enter(TObject *Sender)
{
Memo1->Color = clYellow;
}
```

4. Do zdarzenia **“OnExit”** wprowadźmy kolejny kod:

```
void __fastcall TForm1::Memo1Exit(TObject *Sender)
{
Memo1->Color = clWindow;
}
```

Ćwiczenie 68. PopUpMenu - kopiowanie, wycinanie i wstawianie tekstu do pól typu Edit i Memo

1. Do formularza wstawiamy komponent „**PopUpMenu**", rozbudowujemy go o następujące zdarzenia „Copy", „Cut", „Paste".
2. Na formularzu umieszczamy również dwa pola „**Edit**" oraz dwa pola „**Memo".**
3. Do zdarzenia formularza „**OnCreate**" wprowadzamy kod:

```
void __fastcall TForm1::FormCreate(TObject *Sender)
{
PopupMenu1->AutoPopup = true;
Edit1->PopupMenu = PopupMenu1;
Edit2->PopupMenu = PopupMenu1;
Memo1->PopupMenu = PopupMenu1;
Memo2->PopupMenu = PopupMenu1;
}
```

4. Do zdarzenia "**Copy**" obiektu **PopUpMenu** wprowadzamy kod:

```
void __fastcall TForm1::Copy1Click(TObject *Sender)
{
TComponent *pComponent = PopupMenu1->PopupComponent;
if (pComponent)
{
if (pComponent->ClassNameIs("TEdit"))
((TEdit *)pComponent)->CopyToClipboard();
else if (pComponent->ClassNameIs("TMemo"))
((TMemo *)pComponent)->CopyToClipboard();
else
MessageBeep(0);
}
else
MessageBeep(0);
}
```

5. Do zdarzenia "**Cut**" obiektu **PopUpMenu** wprowadzamy kod:

```
void __fastcall TForm1::Cut1Click(TObject *Sender)
{
TComponent *pComponent = PopupMenu1->PopupComponent;
if (pComponent)
{
if (pComponent->ClassNameIs("TEdit"))
((TEdit *)pComponent)->CutToClipboard();
else if (pComponent->ClassNameIs("TMemo"))
((TMemo *)pComponent)->CutToClipboard();
else
MessageBeep(0);
}
else
MessageBeep(0);
}
```

6. Do zdarzenia "**Paste**" obiektu **PopUpMenu** wprowadzamy kod:

```
void __fastcall TForm1::Paste1Click(TObject *Sender)
{
TComponent *pComponent = PopupMenu1->PopupComponent;
if (pComponent)
{
if (pComponent->ClassNameIs("TEdit"))
((TEdit *)pComponent)->PasteFromClipboard();
else if (pComponent->ClassNameIs("TMemo"))
((TMemo *)pComponent)->PasteFromClipboard();
else
MessageBeep(0);
}
else
MessageBeep(0);
}
```