Mirosław J. Kubiak

ZŁOTE MYŚLI

Programuję w Delphi i C++ Builder

Jak szybko nauczyć się programowania w dwóch różnych językach

Niniejszy **darmowy** ebook zawiera fragment
pełnej wersji pod tytułem:
"Programuję w Delphi i C++ Builder"
Aby przeczytać informacje o pełnej wersji, kliknij tutaj

Darmowa publikacja dostarczona przez
PatBank - bank banków

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

© Copyright for Polish edition by ZloteMysli.pl
Data: 30.09.2006
Tytuł: Programuję w Delphi i C++ Builder (fragment utworu)
Autor: Mirosław J. Kubiak

Projekt okładki: Marzena Osuchowicz
Korekta: Sylwia Fortuna
Skład: Anna Popis-Witkowska

Internetowe Wydawnictwo Złote Myśli
Netina Sp. z o.o.
ul. Daszyńskiego 5
44-100 Gliwice
WWW: www.ZloteMysli.pl
EMAIL: kontakt@zlotemysli.pl

Wszelkie prawa zastrzeżone.
All rights reserved.

SPIS TREŚCI

WSTĘP **6**

ROZDZIAŁ 1. WPROWADZENIE DO PROGRAMOWANIA **11**
- Świat algorytmów 11
- Metody prezentacji algorytmów 13
- Algorytm liniowy a algorytm z rozgałęzieniami 14
- Pseudojęzyk 17
- Na czym polega pisanie programów 17
- Programowanie strukturalne 20
- Programowanie obiektowe 22
- Język programowania Pascal - rys historyczny 22
- Język programowania Delphi – programowanie wizualne 23
- Języki programowania C/C++ - rys historyczny 24
- C++ Builder – programowanie wizualne 25
- Programowanie zdarzeniowe 26
- Warto zapamiętać 27

ROZDZIAŁ 2. TWORZYMY PIERWSZY PROGRAM **30**
- Rozszerzenia plików w Delphi i C++ Builder 30
- Mój pierwszy program 31
- Projekt 37
- Tworzymy prostą aplikację 37
- Menu główne i paski narzędzi 40
- Paleta komponentów 40
- Tworzymy nasz pierwszy program 43
- Program i jego struktura 46
- Moduły w Delphi 48
- Preprocesor 50
- Dyrektywa #include 50
- Dyrektywa #pragma 51
- Moduły w C++ Builder 54
- Tworzymy drugi program 55
- Komentarze 58
- Tworzenie nowych programów i zapisywanie ich na dysku 58
- Wprowadzanie programu źródłowego z dysku do edytora kodu źródłowego 59
- Kończenie pracy w środowisku Delphi (lub C++ Builder) 59
- Warto zapamiętać 59

ROZDZIAŁ 3. DEKLARACJA STAŁYCH I ZMIENNYCH W PROGRAMIE **61**
- Identyfikatory 61
- Deklarujemy stałe w programie 62
- Deklarujemy zmienne w programie 64
- Słowa kluczowe 68
- Nadawanie zmiennym wartości 69
- Warto zapamiętać 79

ROZDZIAŁ 4. OPERACJE WEJŚCIA/WYJŚCIA – CZĘŚĆ I **81**
- Standardowe operacje wejścia/wyjścia 81
- Obsługa sytuacji wyjątkowych 89
- Warto zapamiętać 95

ROZDZIAŁ 5. PROSTE OPERACJE ARYTMETYCZNE **96**
- Podstawowe operatory arytmetyczne 96
- Warto zapamiętać 104

ROZDZIAŁ 6. PODEJMUJEMY DECYZJE W PROGRAMIE **105**
- Podstawowe operatory relacji 105
- Instrukcje warunkowe 106
- Instrukcja warunkowa if 107

Operatory logiczne koniunkcji AND i alternatywy OR....117
Instrukcje wyboru....139
Pierwszy większy program....147
Warto zapamiętać....161

ROZDZIAŁ 7. ITERACJE....163
Instrukcje iteracyjne....163
Instrukcja for....164
Jak działa pętla for?....167
Operatory inkrementacji i dekrementacji....174
Poznajemy operator modulo....179
Poznajemy operator negacji....182
Zmiana przyrostu zmiennej sterującej pętlą....185
Liczby Fibonacciego....204
Analizujemy pozostałe instrukcje iteracyjne....208
Instrukcja iteracyjna repeat (Delphi)....209
Instrukcja iteracyjna do ... while (C++ Builder)....212
Schemat Hornera....215
Algorytm Euklidesa....222
Instrukcja iteracyjna while (Delphi)....228
Instrukcja iteracyjna while (C++ Builder)....230
Priorytety poznanych operatorów....233
Typ wyliczeniowy....234
Warto zapamiętać....239

ZAPROSZENIE DO II CZĘŚCI KSIĄŻKI....242

DODATEK....243
D1. Formatowanie łańcuchów tekstowych....243
D2. Wybrane systemowe procedury i funkcje konwersji typu w Delphi i C++ Builder....245
D3. Wyświetlanie komunikatów....246

Rozdział 2. Tworzymy pierwszy program

W tym rozdziale napiszemy pierwszy program w językach Delphi i w C++ Builder, poznamy zasady pisania programów w językach Delphi i w C++ Builder oraz ćwiczenia podstawowe, jakie powinien opanować każdy, aby biegle pisać programy w edytorze kodów źródłowych, nagrywać je na dysk, wczytywać z dysku programy źródłowe oraz je kompilować i uruchamiać, poznamy również strukturę programu dla języków Delphi i C++ Builder.

Rozszerzenia plików w Delphi i C++ Builder

Z rozdziału 1 wiemy, że w środowisku programistycznym Delphi (i również w C++ Builder) dla każdego projektu jest tworzone wiele plików. Nazwa pliku składa się z dwóch elementów: nazwy nadanej projektowi i jego modułom oraz predefiniowanego rozszerzenia stosowanego przez Delphi (lub C++ Builder). Tabela poniżej przedstawia rozszerzenia plików stosowane w obu środowiskach.

Tabela 2.1. Rozszerzenia plików stosowane w Delphi i w C++ Builder

Element	Delphi	C++ Builder
Plik projektu	*.dpr*	*.bpr*
Plik grupy projektowej	*.bpg*[1]	*.bpg*

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Plik kodu źródłowego	*.pas*	*.cpp*
Plik nagłówkowy	*brak*	*.h lub .hpp*
Plik formularza	*.dfm*	*.dfm*
Skompilowany plik binarny	*.dcu*	*.obj*
Skompilowany plik zasobów	*.res lub .dcr*	*.res*
Zapisanie ustawień pulpitu	*.dsk*	*.dsk*
Ustawienia projektu	*.dof*	*brak*
Pliki źródłowe pakietu	*.dpk*	*.bpk*
Skompilowany pakiet	*.bpl*	*.bpl*
Układ pulpitu	*.dst*	*.dst*

Dla celów naszej książki, zdecydowaliśmy umieścić w odpowiednich katalogach pliki tworzone przez środowisko Delphi (lub C++ Builder). Na przykład w katalogu **P01** znajdują się pliki stworzone przez środowisko Delphi (lub C++ Builder) i związany z nimi określony programistyczny problem. Dodatkowo należy stworzyć katalog dla Delphi oraz dla C++ Builder, gdzie będą umieszczane wszystkie podkatalogi, np. **P01** zawierające określony problem programistyczny dla danego języka programowania.

[1]Łatwo zauważyć, że w niektórych przypadkach rozszerzenia plików stosowane przez środowiska programistyczne Delphi i C++ Builder są takie same.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Mój pierwszy program

Poniżej zamieszczono pierwszy program napisany w języku programowania Delphi i C++ Builder.

Delphi (**katalog P01**):

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Edit1.Text:= 'Moj pierwszy program w Delphi';
end;

end.
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

A oto ten sam program napisany w języku C++ Builder (**katalog P01**):

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Edit1->Text="Moj pierwszy program w C++ Builder";
}
//---------------------------------------------------------------------------
```

Proszę się nie martwić, jeśli instrukcje języka Delphi lub C++ Builder są dla nas zupełnie niezrozumiałe. Ich znaczenie dokładnie poznamy w tej i w następnych lekcjach[2]. Pisząc programy w Delphi (lub w C++ Builderze) będziemy korzystali ze zintegrowanego środowiska Delphi (lub C++ Buildera), które funkcjonuje w systemie operacyjnym *Windows*.

[2]Jak przekonany za chwilę się, wiele linijek kodu generuje samo środowisko Delphi (C++ Builder).

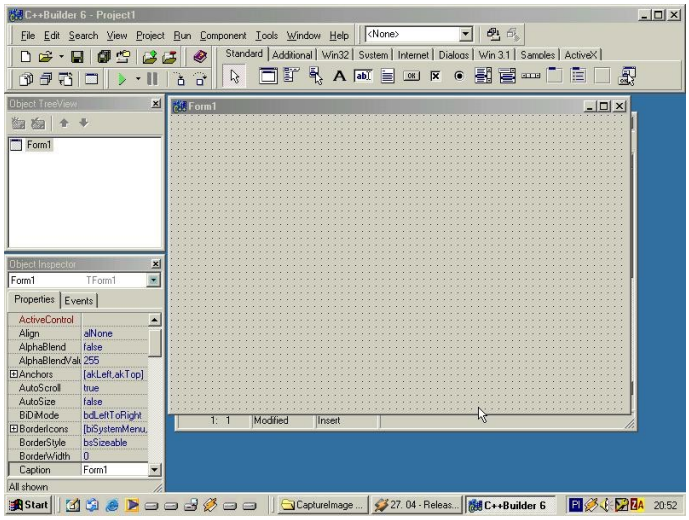
Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Aby dobrze zapoznać się z nowym środowiskiem programistycznym, jakie oferuje Delphi lub C++ Builder, warto dokładnie zapoznać się z jego podstawową filozofią. Języki Turbo Pascal (TP), Turbo C++ i związane z nimi środowisko *DOS* to bardzo proste narzędzia do nauki programowania, gdzie ciężar jego funkcjonowania był usytuowany na pisaniu algorytmów. Pisanie nieskomplikowanych programów wyłącznie do celów dydaktycznych nie sprawiało w tym środowisku żadnych trudności. Wadą tego środowiska był brak gotowych i wygodnych interfejsów komunikacyjnych pomiędzy programem a użytkownikiem.

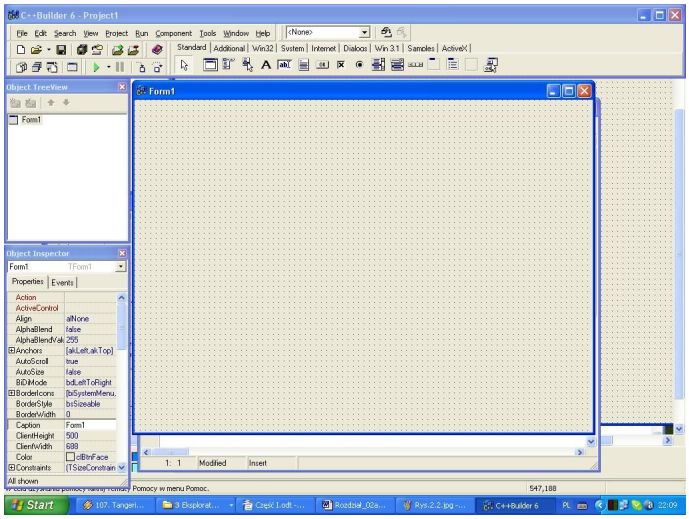
Delphi (i C++ Builder) to wyrafinowane, nowoczesne i wizualne środowisko programistyczne, oferujące wiele gotowych rozwiązań, w którym należy się odnaleźć, aby swobodnie móc w nim pisać proste lub skomplikowane aplikacje.

Autor postawił sobie za zadanie poprowadzenia Czytelnika poprzez oba środowiska programistyczne w sposób prosty, ilustrując naukę programowania w językach Delphi i C++ Builder nieskomplikowanymi przykładami praktycznych programów. Głównym celem tej książki jest przybliżenie obu środowisk programistycznych, bez wnikania w ich zawiłości, aby móc swobodne pisać różne programy w obu językach programowania nie tylko do celów dydaktycznych.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak



Rys. 2.1. Zintegrowane środowisko Delphi z widocznym u góry menu i paletą komponentów. Poniżej na środku widnieje formularz (Form1), a z lewej strony znajduje się Object TreeView oraz Object Inspector. Edytor kodu źródłowego znajduje się pod formularzem.



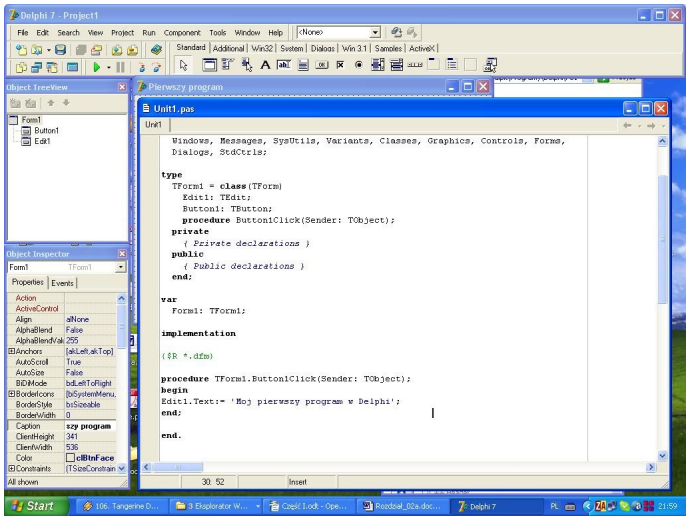
Rys. 2.2. Zintegrowane środowisko C++ Builder z widocznym u góry menu i paletą komponentów. Poniżej na środku widnieje formularz (Form1), a z lewej strony znajduje się Object TreeView oraz Object Inspector. Edytor kodu źródłowego znajduje się pod formularzem.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Zintegrowane środowisko Delphi (C++ Builder) IDE (ang. *Integrated Development Environment*) składa się m.in. z następujących elementów:

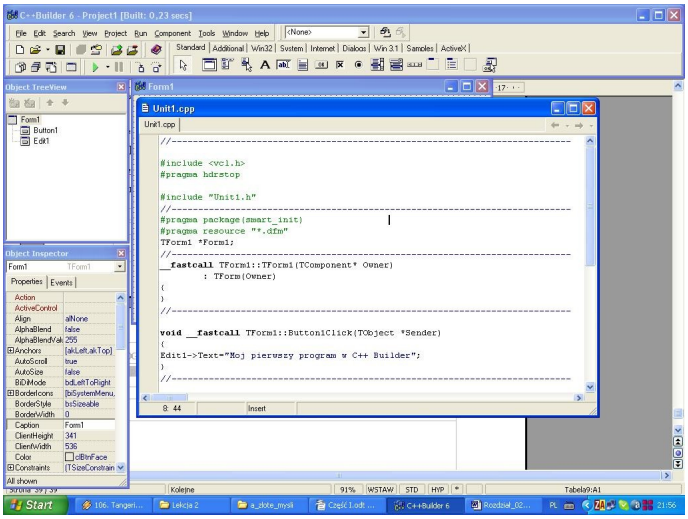
- menu główne i paski narzędzi,
- palety komponentów,
- projektanta formularzy (ang. *Form Designer*),
- edytora kodu źródłowego (ang. *Code Editor*),
- inspektora obiektów (ang. *Object Inspector*) wraz z oknem hierarchii komponentów oraz
- menedżera projektów.

Nie będziemy omawiali dokładnie tych części składowych środowiska Delphi, odsyłając Czytelnika do istniejącej bogatej literatury np. [Reisdorph, 2001, Dorobek, 2003]. Wybierzemy tylko to, co jest nam potrzebne i niezbędne do dalszej nauki pisania programów.



Rys. 2.3. Zintegrowane środowisko Delphi z widocznym na środku edytorem kodu źródłowego, formularz znajduje się pod edytorem.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak



Rys. 2.4. Zintegrowane środowisko C++ Builder z widocznym na środku edytorem kodu źródłowego, formularz znajduje się pod edytorem.

Projekt

Jest to zestaw źródłowych i binarnych plików wzajemnie ze sobą powiązanych, które po procesie kompilacji tworzą jeden wykonywalny program (*.*exe*) lub bibliotekę *.*dll*.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Tworzymy prostą aplikację

Prostą aplikację możemy stworzyć realizując następujące kroki:

1. Utwórz na dysku dowolny katalog, w którym będziesz zapisywał wszystkie pliki tworzone przez Delphi (C++ Builder) dla danego projektu.
2. Uruchom Delphi (C++ Builder) w taki sposób, jak uruchamia się programy w *Windows*.

Na ekranie monitora zobaczysz zintegrowane środowisko Delphi (C++ Builder). Na środku ekranu zobaczysz okno projektanta formularzy (ang. *Form Designer*) o nazwie **Form1**. Pod nim znajduje się edytor kodu źródłowego (ang. *Code Editor*) o nazwie **Unit1.pas** (**Unit1.cpp**). Zawartość edytora kodu, generowana przez program jest następująca dla:

Unit1.pas

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs;

type
  TForm1 = class(TForm)
  private
    { Private declarations }
  public
```

```
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.dfm}

end.
```

Unit1.cpp

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Delphi (C++ Builder) jest tak skonstruowany, że wszelkie zmiany dokonane na formularzu pociągają za sobą zmiany w edytorze kodu, co oznacza że edytor kodu jest ściśle związany z projektantem formularzy. Jeśli teraz porównamy nasz pierwszy program napisany w Delphi (lub C++ Builder), a przedstawiony na początku tej lekcji,

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

to możemy zobaczyć, ile jest naszego wkładu w projekt, a ile samego środowiska Delphi.

4. Wybierz opcję **File|Save All**. Otworzy się okienko z pytaniem o nazwę pliku źródłowego modułu. W polu **Save** (**Zapisz**) wskaż utworzony przed chwilą katalog.
5. W polu **Nazwa pliku** wpisz swoją nazwę (lub pozostaw domyślną) i naciśnij przycisk **Zapisz**.
6. Teraz należy podać nazwę projektu. W polu **Nazwa pliku** wpisz swoją nazwę projektu (lub pozostaw domyślną) i ponownie naciśnij przycisk **Zapisz**.

Podczas swojej pracy Delphi (i C++ Builder) tworzy szereg plików, które zostały omówione wcześniej. Wspomnimy jeszcze tylko o jednym wspólnym rozszerzeniu pliku.

Rozszerzenie	Opis
.exe	Wykonywalny program wynikowy

Menu główne i paski narzędzi

Menu główne zawiera wszystkie opcje niezbędne do pracy, natomiast paski narzędzi zawierają wygodne skróty do często powtarzanych poleceń. Filozofia menu głównego i pasków narzędzi jest intuicyjna i podobna, jak w systemie *Windows*.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Paleta komponentów

Służy do wybierania komponentów albo innych kontrolek (np. kontrolek *ActiveX*) w celu umieszczenia ich na formularzu. Paleta ta składa się z wielu kart, zwanych czasami stronami. Kliknięcie odpowiedniej zakładki powoduje otwarcie odpowiadającej jej karty. W tej książce będziemy się posługiwali m.in. kartami *Standard* i *Additional*.

Rys. 2.3. Paleta komponentów – karta Standard.

Rys. 2.4. Paleta komponentów – karta Additional.

Proces umieszczania komponentu na formularzu odbywa się w dwóch etapach:

1. wybranie odpowiedniego komponentu znajdującego się na palecie kliknięciem myszy,
2. umieszczenie tego komponentu na formularzu kliknięciem myszy.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Oto następujące komponenty, które zostały wykorzystane w tej książce:

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Karta Standard

Komponent	Nazwa komponentu
	Edit – liniowe pole tekstowe, służące do np. pobrania danych od użytkownika
	Label – etykieta służąca do wyświetlania tekstu
	Button – przycisk
	Panel – pozwala na zamieszczanie innych komponentów
	ListBox – umożliwia zgrupowanie listy elementów
	MainMenu – pozwala na utworzenie w prosty sposób menu dla własnej aplikacji
	Memo – pole tekstowe służące do edycji dużej ilości tekstu

Karta Additional

Komponent	Nazwa komponentu
	StringGrid – pozwala na umieszczeniu na formularzu siatki, składającej się z kolumn i wierszy

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Tworzymy nasz pierwszy program

Delphi

1. Postępujemy dokładnie jak w punkcie **Tworzymy prostą aplikację**.
2. Następnie na palecie komponentów klikamy myszą przycisk **Button** i ponownie klikamy myszą na formularzu w miejscu, gdzie chcemy, żeby się on znajdował. Następnie w narzędziach *Object Inspectora* szukamy *Caption* i zamiast **Button1** wpisujemy słowo **Zacznij**. *Object Inspector* pokazuje właściwości (*Properties)* i zdarzenia (*Events*), które zostały przypisane różnym komponentom, dając łatwą możliwość zmiany ich właściwości.
3. Z palety komponentów wybieramy komponent **Edit** i umieszczamy go na formularzu. W *Object Inspectorze* szukamy *Text* i usuwamy słowo **Edit1**.
4. Dwukrotnie klikamy na formularzu to miejsce, gdzie został umieszczony komponent **Zacznij**, aby obsłużyć związane z tym komponentem zdarzenie. W okienku **Unit1.pas** ukazał się dodatkowy fragment kodu widoczny w ramce poniżej:

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

end;

end.
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Pomiędzy słowami kluczowymi **begin end;** wpisujemy zasadniczy fragment kodu naszego programu: **Edit1.Text:= 'Moj pierwszy program w Delphi';**

5. Naciskając klawisz **F9** lub **Run|Run**, lub przycisk kompilujemy nasz program.
6. Po skompilowaniu na formularzu naciskamy przycisk **Zacznij**, który wygenerował tekst: **Moj pierwszy program w Delphi**.
7. Powtarzamy wszystkie czynności, aby je utrwalić.

C++ Builder

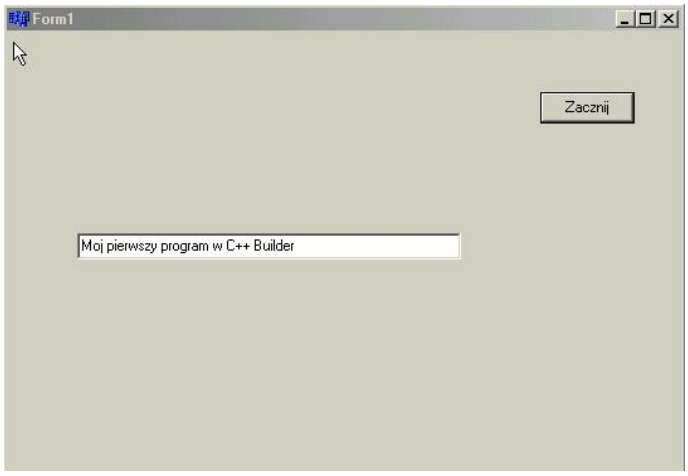
1. Postępujemy dokładnie jak w punkcie **Tworzymy prostą aplikację**.
2. Następnie na palecie komponentów klikamy myszą przycisk **Button** i ponownie klikamy myszą na formularzu w miejscu, gdzie chcemy, żeby się on znajdował. Następnie w narzędziach *Object Inspectora* szukamy *Caption* i zamiast **Button1** wpisujemy słowo **Zacznij**. *Object Inspector* pokazuje właściwości (*Properties*) i zdarzenia (*Events*), które zostały przypisane różnym komponentom, dając łatwą możliwość zmiany ich właściwości.
3. Z palety komponentów wybieramy komponent **Edit** i umieszczamy go na formularzu. W *Object Inspectorze* szukamy *Text* i usuwamy słowo **Edit1**.
4. Dwukrotnie klikamy na formularzu to miejsce, gdzie został umieszczony komponent **Zacznij**, aby obsłużyć związane

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

z tym komponentem zdarzenie. W okienku **Unit1.cpp** ukazał się dodatkowy fragment kodu widoczny w ramce poniżej:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{

}
```

5. Pomiędzy nawiasami klamrowymi **{ };** wpisujemy zasadniczy fragment kodu naszego programu: **Edit1->Text="Moj pierwszy program w C++ Builder";**
6. Naciskając klawisz **F9** lub **Run|Run**, lub przycisk kompilujemy nasz program.
7. Po skompilowaniu na formularzu naciskamy przycisk **Zacznij**, który wygenerował tekst: **Moj pierwszy program w C++ Builder**.
8. Powtarzamy wszystkie czynności, aby je utrwalić.



Rys. 2.7. Mój pierwszy program w C++ Builderze (analogiczny rezultat otrzymamy w Delphi) – katalog (P01).

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Program i jego struktura

Aplikacje w zintegrowanym środowisku Delphi (i C++ Builder) są zbudowane z modułów. W środowisku Delphi (i C++ Builder) każdy formularz jest związany z jakimś modułem, który definiuję klasę[3] **Tform1** dla tego formularza.

Delphi

Aplikację w środowisku Delphi tworzą pliki o dwóch rodzajach kodu źródłowego:

1. główny plik programu oraz
2. jeden lub więcej modułów.

Poniżej zamieszczono strukturę głównego pliku programu.

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

[3]Więcej o klasach Czytelnik dowie się w rozdziale 12.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Na początku programu znajduje się *słowo kluczowe*[4] **program**, po którym następuje nazwa programu np. **Project1** zakończona średnikiem. Deklaracja **uses** specyfikuje listę modułów składających się na program. Moduły w Delphi to pliki drugorzędne, do których odwołujemy się poprzez program. Ogromną zaletą środowiska Delphi jest automatyczne generowanie pliku projektu. Aby się o tym przekonać, należy uruchomić Delphi, a następnie kliknąć **Project| View Source**.

Sekcja **uses** zawiera moduł **Forms** oraz moduł **Unit1** z dyrektywą **in** specyfikującą nazwę pliku zawierającą podany moduł. Zapis **Unit1 in 'Unit1.pas'** oznacza, że moduł **Unit1** znajduje się w pliku **Unit1.pas**. Słowo kluczowe **uses** umożliwia zarządzanie kompilacją i konsolidacją aplikacji.

Główny kod tej aplikacji:

```
Application.Initialize;
Application.CreateForm(TForm1, Form1);
Application.Run;
```

znajduje się pomiędzy słowami kluczowymi **begin** i **end**. Zarówno moduł, jak i program musi zostać zakończony kropką.

[4]Słowa kluczowe zostaną omówione w rozdziale 3.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Moduły w Delphi

Każdy formularz został zdefiniowany w module. Moduły umożliwiają definiowanie i udostępnianie zestawu podprogramów, czyli procedur i funkcji. Pusty moduł, zawierający kod poniżej, uzyskamy naciskając **File|New|Unit**.

```
unit Unit1;
interface
implementation
end.
```

Moduł, który ma swoją jednoznaczną nazwę - w tym wypadku (**Unit1**) - składa z sekcji **interface**, gdzie deklaruje się wszystkie identyfikatory widoczne dla innych modułów oraz z sekcji **implementation** zawierającej kod programu i dodatkowe deklaracje.

W module powinny zostać zadeklarowane wszystkie identyfikatory przed ich użyciem. W sekcji **interface** oraz w sekcji **implementation** mogą być zadeklarowane:

- stałe za pomocą słowa kluczowego **const**,
- zmienne i obiekty za pomocą słowa kluczowego **var**,
- typy i klasy za pomocą słowa kluczowego **type**,
- funkcje i procedury.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

C++ Builder

Poniżej przedstawiono strukturę głównego pliku programu.

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop
//---------------------------------------------------------------------------
USEFORM("Unit1.cpp", Form1);
//---------------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
	try
	{
		Application->Initialize();
		Application->CreateForm(__classid(TForm1),
&Form1);
		Application->Run();
	}
	catch (Exception &exception)
	{
		Application->ShowException(&exception);
	}
	catch (...)
	{
		try
		{
			throw Exception("");
		}
		catch (Exception &exception)
		{
			Application->ShowException(&exception);
		}
	}
	return 0;
}
//---------------------------------------------------------------------------
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Preprocesor

Kod programu znajdujący się w plikach źródłowych nie jest poddawany kompilacji w postaci, w jakiej możemy zobaczyć w edytorze. Wersję plików źródłowych przeznaczoną do kompilacji generuje mechanizm zwany *preprocesorem*. Przegląda on kod w poszukiwaniu instrukcji, które ma wykonać. Tymi instrukcjami są *dyrektywy* preprocesora, które rozpoczynają się od znaku #. Efektem pracy preprocesora jest zmodyfikowana postać kodu źródłowego. Modyfikacje wykonane przez preprocesor w kodzie programu nie są widoczne i istnieją jedynie podczas kompilacji.

Na początku głównego pliku programu znajdują się dwie dyrektywy preprocesora:

#include <vcl.h>

#pragma hdrstop

Poniżej pokrótce omówimy ich znaczenie.

Dyrektywa #include

Jej zadaniem jest dołączenie do pliku o podanej nazwie do kodu źródłowego. Jest ona najczęściej wykorzystywana do włączania *plików nagłówkowych* (*.h). Ogólna jej postać jest następująca:

#include <nazwa_pliku>.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Aby plik mógł być dołączony, musi się on znajdować w ścieżce plików nagłówkowych. Zapis

#include "nazwa_pliku"

oznacza, że plik jest najpierw poszukiwany w katalogu bieżącym (tam, gdzie znajduje się plik źródłowy projektu), a następnie w ścieżce plików nagłówkowych.

Dyrektywa #pragma

Służy do definiowania nowych dyrektyw, specyficznych dla danego kompilatora. Ta dyrektywa kończy listę statycznych plików nagłówkowych (włączanych za pomocą dyrektywy **include**).

Dyrektywa **#pragma hdrstop** (ang. *header stop*) informuje kompilator, że właśnie nastąpił koniec listy plików nagłówkowych, które mają być prekompilowane.

Dyrektywa **#include <vcl.h>** oznacza, że do programu należy dołączyć pliki nagłówkowe biblioteki VCL.

Każdy program w C, C++ oraz w C++ Builder musi zawierać w sobie przynajmniej jedną funkcję. Funkcja **WinMain()** jest główną funkcją programu C++ Builder[5] i każdy program ma dokładnie jedną taką funkcję, nawet jeśli składa się z wielu modułów. Jest ona

[5]Programy pisane w C++ Builder i posługujące się klasą formularza nie zawierają funkcji **void main(void)** znaną z języka C/C++ .

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

wywoływana jako pierwsza i zawiera w sobie zestaw kolejnych instrukcji wykonywanych przez program, zawartych pomiędzy parą nawiasów klamrowych **{ ... }**.

Składnia funkcji **WinMain()** jest następująca:

WINAPI WinMain(HINSTANCE AktEgz, HINSTANCE PoprzEgz, LPSTR Parametry, int Stan),

gdzie **HINSTANCE** w wolnym tłumaczeniu oznacza uchwyty. Są one niezbędne, gdyż system Windows jest systemem wielozadaniowym, co oznacza, że w danej chwili może działać wiele egzemplarzy danego programu. Argumenty tej funkcji mają następujące znaczenie:

AktEgz – uchwyt aktualnego egzemplarza programu,

PoprzEgz – uchwyt poprzedniego egzemplarza programu,

Parametry – łańcuch z parametrami wywołania programu,

Stan – określa, w jakim stanie jest program.

Główny kod tej aplikacji ma postać:

```
Application->Initialize();
Application->CreateForm(__classid(TForm1), &Form1);
Application->Run();
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Składa się on z procesu inicjacji – metoda[6] **Initialize()**, tworzenia formularza – metoda **CreateForm()** oraz metoda **Run()**. Całość głównego kodu aplikacji umieszczono w nawiasach klamrowych

```
try
 {
  Application->Initialize();
  Application->CreateForm(__classid(TForm1), &Form1);
  Application->Run();
 }
catch (Exception &exception)
 {
  Application->ShowException(&exception);
 }
```

gdzie wykorzystano obsługę sytuacji wyjątkowych **try ... catch**.

Jeśli powyższy proces tworzenia aplikacji się nie powiedzie, to na ekranie zobaczymy komunikat w postaci wygenerowanego przez system tzw. wyjątku (ang. *exception*), wyświetlonego przy pomocy funkcji **ShowException()**.

Instrukcja **return 0** zwraca do systemu operacyjnego kod zakończenia działania programu.

[6]O metodach Czytelnik dowie się więcej w rozdziale12.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Moduły w C++ Builder

Każdy formularz został zdefiniowany w module. Moduły umożliwiają definiowanie i udostępnianie zestawu podprogramów, czyli procedur i funkcji. Pusty moduł, zawierający kod poniżej, uzyskamy naciskając **File|New|Unit**.

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
    : TForm(Owner)
{
}
//---------------------------------------------------------------------------
```

Moduł, który ma swoją jednoznaczną nazwę – w tym wypadku (**Unit1.cpp**) – składa się m.in. z dyrektyw dla preprocesora oraz funkcji **__fastcall**.

Zanim zaczniemy korzystać z formularza, który jest pierwszym obiektem, musi on zostać odpowiednio zainicjowany. Umożliwia to nam specjalna funkcja składowa, nosząca taką samą nazwę jak klasa,

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

do której ten obiekt należy. Prototyp tej funkcji (nazywanej konstruktorem) z parametrami ma postać:

__fastcall TForm1::TForm1(TComponent* Owner) : TForm(Owner)[7]

Tworzymy drugi program

Delphi

Wzbogaceni o dodatkowe informacje spróbujmy napisać program w Delphi, który wypisuje komunikat:

Moj pierwszy program w Delphi nie sprawil mi zadnych trudnosci.

Jeśli mamy nadal trudności z napisaniem takiego programu, to kompletny jego wydruk znajduje się poniżej.

```
unit Unit1;

interface

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

[7]Konwencja **__fastcall** (szybkie wywołanie) zapewnia, że parametry konstruktora zostaną przekazane przez rejestry procesora, a pełny tekst konstruktora klasy **TForm1** zostanie automatycznie umieszczony w module **Unit1.cpp** i tam też zostanie zainicjowany.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

```
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Edit1: TEdit;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
Edit1.Text:='Moj pierwszy program nie sprawil mi zadnych
trudnosci';
end;

end.
```

C++ Builder

Wzbogaceni o dodatkowe informacje dotyczące struktury programu spróbujmy napisać program w C++ Builder, który wypisuje na ekranie komunikat:

Moj pierwszy program w C++ Builder nie sprawil mi zadnych trudnosci.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Jeśli mamy nadal trudności z napisaniem takiego programu, to kompletny jego wydruk znajduje się poniżej.

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
Edit1->Text="Moj pierwszy program nie sprawil mi
zadnych trudnosci";
}
//---------------------------------------------------------------------------
```

Domyślnie generowana nazwa funkcji **Button1Click** obsługi zdarzenia (ang. *event handler*) składa się z nazwy obiektu **Button1** i nazwy zdarzenia **Click**. Natomiast **(TObject *Sender)** to argument funkcji. **TObject** to klasa macierzysta, natomiast ***Sender** to wskaźnik (ang. *pointer*) obiektu klasy **TObject**. Składnia ta opiera się na mechanizmie *dziedziczenia*[8], a program C+

[8]Dziedziczenie zostanie omówione w rozdziale 12.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

+ Builder generuje to automatycznie.

Komentarze

W Delphi (i w C++ Builderze) komentarze możemy wstawiać do programu na trzy sposoby:

{To jest komentarz}

(*To tez jest komentarz*)

//I to tez jest komentarz, ale tylko do konca linii

Tworzenie nowych programów i zapisywanie ich na dysku

Aby utworzyć nową aplikację należy nacisnąć **File|New|Application**. Zapisujemy ją na dysku klikając: **File|Save** lub **File|Save As** lub **File|Save Project As** lub **File|Save All**.

Wprowadzanie programu źródłowego z dysku do edytora kodu źródłowego

Aby wprowadzić programu źródłowego z dysku do edytora kodu źródłowego należy dwukrotnie kliknąć na plik np. **Project1**, który

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

zawiera kod źródłowy projektu. Resztę za nas zrobi środowisko *Windows*.

Kończenie pracy w środowisku Delphi (lub C++ Builder)

Ze środowiska Delphi (lub C++ Builder) wychodzimy naciskając **File|Exit**.

Warto zapamiętać

Podczas tej lekcji nauczyliśmy się pisać proste programy w edytorze tekstów Delphi (i C++ Builder), zapisywać je do pamięci zewnętrznej (dysk twardy, dyskietka) oraz wczytywać wcześniej zapisane pliki z pamięci zewnętrznej do edytora.

1. Wszystkie programy w języku w Delphi rozpoczynają się od słowa kluczowego **program**, po którym następuje nazwa programu.
2. Nazwa programu nie może rozpoczynać się liczbą.
3. Główna część programu w języku C++ Builder rozpoczyna się od instrukcji **WinMain()**.
4. Komentarze w programach będziemy umieszczali w nawiasach **{}** lub używając znaków //.
5. Aby poprawić czytelność programów źródłowych, należy oprócz komentarzy wstawić puste linie rozdzielające

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

niepowiązane ze sobą instrukcje. Gdy kompilator trafi na pustą linię, to po prostu ja pominie.

6. Wszystkie programy (zawarte w tej książce) i napisane w języku Delphi (lub C++ Builder) wyświetlają informacje na ekranie przy pomocy komponentów.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Rozdział 3. Deklaracja stałych i zmiennych w programie

W tym rozdziale dowiemy się, co to są identyfikatory, jak tworzyć i wykorzystywać stałe i zmienne w programach napisanych w języku Delphi (oraz w C++ Builder), poznamy typy zmiennych oraz słowa kluczowe.

Identyfikatory

Programy przechowują informacje w postaci *stałych* i *zmiennych*. Jeśli chcemy umieścić w programie stałe lub zmienne, to musimy je wcześniej zadeklarować. W zależności od tego, jakiego rodzaju wartości chcemy przechowywać, na przykład liczbę całkowitą, liczbę zmiennoprzecinkową lub literę alfabetu, to musimy używać stałych i zmiennych określonego typu. Typ stałych i zmiennych określa typ wartości, jakie mogą one przechowywać, jak również zbiór operacji matematycznych (dodawanie, mnożenie itd.), jakie może program wykonywać na tych wartościach. W czasie tej lekcji przedstawimy, jak tworzyć i wykorzystywać stałe i zmienne w programach napisanych w Delphi (i w C++ Builder).

Do oznaczania nazw stałych, zmiennych, funkcji i innych obiektów zdefiniowanych przez użytkownika będziemy posługiwali się *identyfikatorami*. Są to określone ciągi znaków alfabetu

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

angielskiego, przy czym pierwszy musi znak być literą. Nie wolno definiować identyfikatorów takich samych jak słowa kluczowe. A oto kilka przykładowych identyfikatorów:

Poprawne	Niepoprawne
dlugosc	570dlugosc
minimum	?!minimum
maksimum_boku_a	maksimum boku a

Deklarujemy stałe w programie

Stałe języka Delphi (i C++ Builder) reprezentują ustalone wartości, które nie mogą zostać zmienione podczas działania programu. Mogą być one dowolnego typu.

Delphi

Do deklarowania stałych w języku Delphi służy słowo kluczowe **const** (ang. *constant* - stała). Ogólna postać deklaracji stałej dla tego języka jest następująca:

const

nazwa_stałej = wartość;

Np. deklaracja:

const

cyfra = 10;

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

oznacza, że w całym programie została zadeklarowana stała o nazwie **cyfra** i o wartości równej 10.

C++ Builder

Do deklarowania stałych służy słowo kluczowe **const** (pisane wyłącznie małymi literami). Ogólna postać deklaracji stałej dla tego języka jest następująca:

const typ nazwa_stałej = wartość;

Zapis

const int cyfra = 10;

oznacza, że w całym programie została zadeklarowana stała typu całkowitego **int** (ang. *integer*) o nazwie **cyfra** i o wartości równej 10.

Wartość tak zdefiniowanej stałej w obu językach pozostaje niezmienna w programie, a każda próba zmiany tej wartości będzie sygnalizowana jako błąd.

C++ Builder

Do deklarowania stałych służy słowo kluczowe **const** (pisane wyłącznie małymi literami). Ogólna postać deklaracji stałej dla tego języka jest następująca:

const typ nazwa_stałej = wartość;

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Zapis

const int cyfra = 10;

oznacza, że w całym programie została zadeklarowana stała typu całkowitego **int** (ang. *integer*) o nazwie **cyfra** i o wartości równej 10.

Wartość tak zdefiniowanej stałej w obu językach pozostaje niezmienna w programie, a każda próba zmiany tej wartości będzie sygnalizowana jako błąd.

Deklarujemy zmienne w programie

W językach Delphi i C++ Builder wszystkie zmienne, zanim zostaną użyte w programie, muszą zostać wcześniej zadeklarowane.

Delphi

W języku programowania Delphi musimy w bloku deklaracyjnym zadeklarować zmienne używając słowa kluczowego **var** (ang. *variable* - zmienna). Następnie musimy określić typy zmiennych. Ogólną postać deklaracji zmiennej przestawiono poniżej:

var

nazwa_zmiennej : typ zmiennej;

Na przykład deklaracje zmiennej o nazwie **liczba** typu **integer** deklarujemy w sposób następujący:

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

var

liczba : **integer;**

W języku Delphi (i w C++ Builder) nazwę zmiennej (jej identyfikator) ustala programista.

UWAGA: nazwa zmiennej nie może zaczynać się od cyfry. Pisząc program programista może skorzystać z (zamieszczonych poniżej) następujących predefiniowanych typów:

Delphi

Niektóre typy danych są ze znakiem, a niektóre bez znaku. Zmienne ze znakiem przechowują zarówno liczby dodatnie, jak i ujemne. Zmienne bez znaku przechowują tylko liczby dodatnie.

Tabela 3.1. Wybrane predefiniowane typy w języku Delphi dla aplikacji 32-bitowych.

Typ danej	Rozmiar w bajtach	Zakres wartości
ShortInt	1	-128 – 127
Byte	1	0 – 255
Char	1	0 – 255
WideChar	2	0 – 65 535
SmallInt	2	- 32 768 – 32 767
Word	2	0 – 65 535
LongInt	4	- 2 147 483 648 – 2 147 483 647
Int64	8	od - 9 223 372 036 854 775 808

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

		do 9 223 372 036 854 775 807
Integer	4	Tak samo jak **LongInt**
Cardinal	4	0 – 4 294 967 925
LongWord	4	Tak samo jak **Cardinal**
Single	4	$1,5 \times 10^{-45} – 3,4 \times 10^{38}$
Double	8	$5,0 \times 10^{-324} – 1,7 \times 10^{308}$
Real	8	Tak samo jak **Double**
Extended	10	$3,4 \times 10^{-4932} – 1,1 \times 10^{4932}$
Comp	8	od - 9 223 372 036 854 775 808
		do 9 223 372 036 854 775 807
Boolean	1	**False** lub **True**

C++ Builder

W języku w C++ Builder wszystkie zmienne, zanim zostaną użyte w programie, muszą wcześniej zostać zadeklarowane.

Aby zadeklarować zmienną w programie, musimy określić jej typ i nazwę (identyfikator), przez którą program będzie odwoływał się do tej zmiennej. Ogólną postać deklaracji zmiennej przestawiono poniżej:

typ_zmiennej nazwa zmiennej;

Na przykład deklaracje zmiennej o nazwie **liczba** typu **int** deklarujemy w sposób następujący:

int liczba;

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

W Tabeli 3.2 zawarto najbardziej rozpowszechnione (standardowe) typy zmiennych stosowane w języku w C++ Builder.

Tabela 3.2. Powszechnie stosowane typy zmiennych całkowite i rzeczywiste stosowane w języku w C++ Builder oraz przydzielona dla nich pamięć w bitach.

Typ całkowity	Liczba bitów	Przechowywane wartości
unsigned char	8	0 – 255
signed char	8	- 128 – 127
unsigned int	16	0 – 65 535
short signed int	16	- 32 768 – 32 767
signed int	16	- 32 768 – 32 767
long unsigned int	32	0 – 4 294 967 295
long signed int	32	- 2 147 483 648 – 2 147 483 647

Typ rzeczywisty	Liczba bitów	Przechowywane wartości
float (liczby rzeczywiste)	32	$3.4*10^{-38}$ – $3.4*10^{38}$
double (liczby o podwójnej precyzji)	64	$1.7*10^{-308}$–$1.7*10^{308}$
long double (liczby o wysokiej precyzji)	80	$3.4*10^{-4932}$ –$1.1*10^{4932}$

W języku C++ Builder mamy do dyspozycji typ logiczny **bool**. Jego zmienne mogą przyjmować dwie wartości **true** (prawda) oraz **false** (fałsz).

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Słowa kluczowe

Podczas tworzenia nazw zmiennych musimy pamiętać, że w obu językach programowania istnieją pewne zarezerwowane słowa, które nie mogą być używane jako zmienne. Te zarezerwowane słowa to są słowa kluczowe, które mają specjalne znaczenie dla kompilatora.

Delphi

UWAGA: słowa kluczowe w języku Delphi możemy pisać zarówno dużymi, jak i małymi literami. Umawiamy się, że w tej książce słowa kluczowe w programach napisanych w języku Delphi piszemy małymi literami.

Tabela 3.3. Wybrane słowa kluczowe w języku Delphi.

and	**array**	**asm**	**begin**	**case**
class	**const**	**construct-or**	**destructor**	**div**
do	**downto**	**else**	**end**	**file**
for	**function**	**goto**	**if**	**implemen-tation**
in	**inherited**	**inline**	**interface**	**label**
mod	**nil**	**not**	**of**	**or**
procedure	**program**	**record**	**repeat**	**set**
shl	**shr**	**string**	**then**	**to**
type	**unit**	**until**	**uses**	**var**
while	**with**	**xor**		

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

C++ Builder

UWAGA: słowa kluczowe w języku C++ Builder piszemy wyłącznie małymi literami.

Tabela 3.4. Wybrane słowa kluczowe w C++ Builder.

asm	**auto**	**break**	**case**	**catch**
char	**class**	**const**	**continue**	**default**
delete	**do**	**double**	**else**	**enum**
extern	**float**	**for**	**friend**	**goto**
if	**inline**	**int**	**long**	**new**
operator	**private**	**protected**	**public**	**register**
return	**short**	**signed**	**sizeof**	**static**
struct	**switch**	**template**	**this**	**throw**
try	**typedef**	**union**	**unsigned**	**virtual**
void	**volatile**	**while**		

Nadawanie zmiennym wartości

Jak wcześniej zostało powiedziane, zmienne przechowują wartości w trakcie działania programu.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Delphi

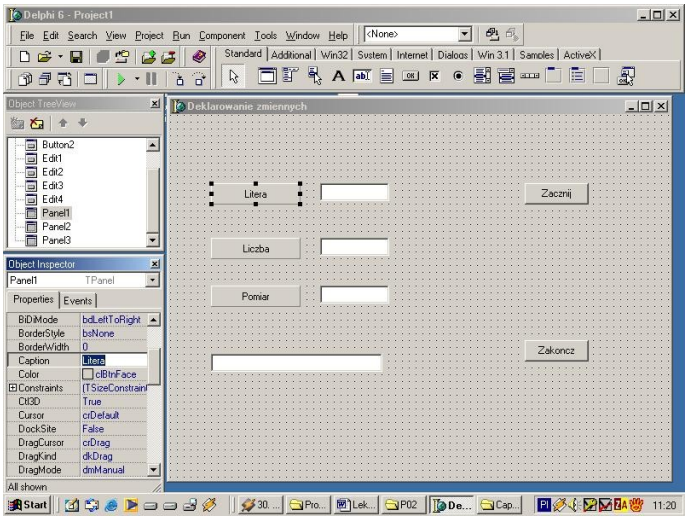
Do przypisania zmiennej nowej wartości służy instrukcja przypisania oznaczona symbolem **:=** (dwukropek i znak równości). Ogólna postać tej instrukcji jest następująca:

nazwa_zmiennej := wyrażenie

Do wyświetlania na ekranie wartości zmiennych posłużymy się np. polem **Edit**. Na formularzu wykonujemy następujące czynności:

1. umieszczamy na nim trzy komponenty **Panel**, cztery komponenty **Edit** oraz dwa przyciski **Button**,
2. po umieszczeniu na formularzu komponentu **Panel1** należy na w *Object Inspectorze* zmienić we właściwościach (*Properties*) w *Caption* **Panel1** na **Litera**, **Panel2** na **Liczba** oraz **Panel3** na **Pomiar**,
3. po umieszczeniu na formularzu przycisku **Button** należy w *Object Inspectorze* zmienić we właściwościach (*Properties*) w *Caption* **Button1** na **Zacznij** i **Button2** na **Zakończ**. Ilustruje to rysunek poniżej.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak



Rys. 3.1. Program P02 podczas projektowania formularza.

4. następnie dwukrotnie klikamy na formularzu przycisk **Zacznij** i podpinamy związaną z nim procedurę procedurę **TForm1.Button1Click(Sender: TObject)** z następującym kodem:

```
procedure TForm1.Button1Click(Sender: TObject);
var
 litera : Char;
 liczba : Integer;
 pomiar : Real;
begin
 litera:='A';
 liczba:=120;
 pomiar:=78.231;
```

```
 Edit1.Text:=litera;
 Edit2.Text:=IntToStr(liczba);
 Edit3.Text:=Format('%2.2f', [pomiar]);
end;
```

5. W ostatnim etapie dwukrotnie klikamy przycisk **Zakończ** i w procedurze
procedure TForm1.Button2Click(Sender: TObject) umieszczamy następujący kod:

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 Edit4.Text:='Koniec programu';
end;
```

Kilka słów wyjaśnienia poświęcimy następującym linijkom kodu:

Edit2.Text:=IntToStr(liczba);

Edit3.Text:=Format('%2.2f', [pomiar]);

Zapis **Edit2.Text:=InttoStr(liczba)** oznacza, że aby w polu **Edit2** mogła się pojawić zmienna **liczba**, to musi na niej zostać wykonana konwersja typu przy pomocy procedury **IntToStr**. Więcej informacji na temat systemowych procedur konwersji typu znajdzie czytelnik w Dodatku na końcu książki.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Zapis **Edit3.Text:=Format('%2.2f', [pomiar])** oznacza, że aby w polu **Edit3** mogła się pojawić zmienna **pomiar** typu rzeczywistego, musi ona zostać najpierw poddana formatowaniu. Więcej informacji na temat formatowania łańcuchów tekstowych znajdzie Czytelnik w Dodatku.

Poniżej znajduje się cały program napisany w Delphi (**P02**).

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes,
Graphics, Controls, Forms,
  Dialogs, ExtCtrls, StdCtrls;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Button1: TButton;
    Button2: TButton;
    Panel3: TPanel;
    Edit4: TEdit;
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

```
implementation

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var
litera : Char;
liczba : Integer;
pomiar : Real;

begin
litera:='A';
liczba:=120;
pomiar:=78.231;

Edit1.Text:=litera;
Edit2.Text:=IntToStr(liczba);
Edit3.Text:=Format('%2.2f', [pomiar]);
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
Edit4.Text:='Koniec programu';
end;

end.
```

C++ Builder

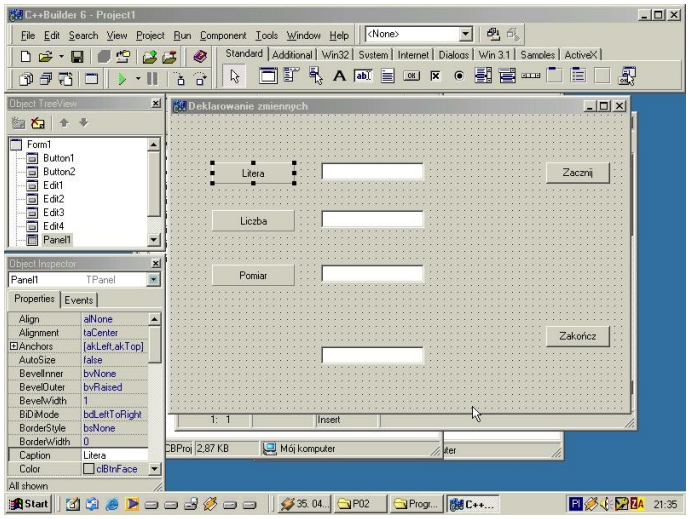
Po zadeklarowaniu zmiennej, przy użyciu operatora przypisania (oznaczonego znakiem równości =) możemy przypisać jej odpowiednią wartość. Ogólna postać instrukcji przypisania jest następująca:

nazwa_zmiennej = wyrażenie

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Do wyświetlania na ekranie wartości zmiennych posłużymy się polem **Edit**. Na formularzu wykonujemy następujące czynności:

1. umieszczamy na nim trzy komponenty **Panel**, cztery komponenty **Edit** oraz dwa przyciski **Button**,
2. po umieszczeniu na formularzu komponentu **Panel1** należy na w *Object Inspectorze* zmienić we właściwościach (*Properties*) w *Caption* **Panel1** na **Litera**, **Panel2** na **Liczba** oraz **Panel3** na **Pomiar**.
3. po umieszczeniu na formularzu przycisku **Button** należy w *Object Inspectorze* zmienić we właściwościach (*Properties*) w *Caption* **Button1** na **Zacznij** i **Button2** na **Zakończ**. Ilustruje to rysunek poniżej.



Rys. 3.2. Program P02 podczas projektowania formularza.

4. następnie dwukrotnie klikamy na formularzu przycisk **Zacznij** i podpinamy związaną z nim funkcję **void __fastcall**

TForm1::Button1Click(TObject *Sender) z następującym kodem:

```
void __fastcall TForm1::Button1Click(TObject
*Sender)
{
 char litera;
 int liczba;
 float pomiar;
 litera = 'A';
 liczba = 120;
 pomiar = 78.231;
 Edit1->Text=litera;
 Edit2->Text=IntToStr(liczba);
 Edit3->Text=FloatToStrF(pomiar,ffNumber,4,2);
}
```

5. W ostatnim etapie dwukrotnie klikamy przycisk **Zakończ** i w funkcji **void __fastcall Tform1::Button2Click (TObject *Sender)** umieszczamy następujący kod:

```
void __fastcall TForm1::Button2Click(TObject
*Sender)
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

```
{
 Edit4->Text="Koniec programu";
}
```

Kilka słów wyjaśnienia poświęcimy następującym linijkom kodu:

```
Edit2->Text=IntToStr(liczba);
Edit3->Text=FloatToStrF(pomiar,ffNumber,4,2);
```

Zapis **Edit2->Text=IntToStr(liczba)** oznacza, że aby w polu **Edit2** mogła się pojawić zmienna **liczba**, to musi na niej zostać wykonana konwersja typu przy pomocy procedury **IntToStr**. Więcej informacji na temat systemowych procedur konwersji typu znajdzie czytelnik w Dodatku.

Zapis **Edit3->Text=FloatToStrF(pomiar,ffNumber,4,2)** oznacza, że aby w polu **Edit3** mogła się pojawić zmienna **pomiar** typu rzeczywistego, musi ona zostać poddana najpierw „formatowaniu". Więcej informacji na temat „formatowania" łańcuchów tekstowych znajdzie Czytelnik w Dodatku na końcu książki.

Poniżej znajduje się cały program napisany w C++ Builder (**P02**).

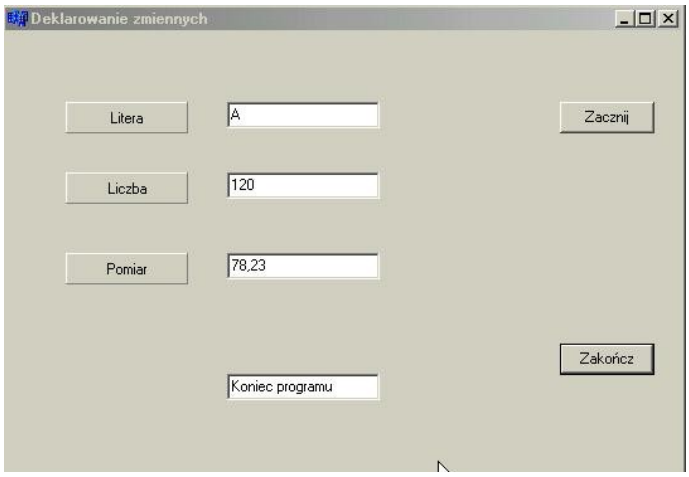
```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

```
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::Button1Click(TObject *Sender)
{
 char litera;
 int liczba;
 float pomiar;

 litera = 'A';
 liczba = 120;
 pomiar = 78.231;

 Edit1->Text=litera;
 Edit2->Text=IntToStr(liczba);
 Edit3->Text=FloatToStrF(pomiar,ffNumber,4,2);
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button2Click(TObject *Sender)
{
 Edit4->Text="Koniec programu";
}
//---------------------------------------------------------------------------
```

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Po skompilowaniu obu programów naciskamy przycisk **Zacznij.**



Rys. 3.3. Efekt działania programu P02 w obu językach programowania.

Warto zapamiętać

1. Identyfikator to ciąg znaków, z których pierwszy musi być literą. Nie można definiować identyfikatorów takich samych jak słowa kluczowe.
2. Słowa kluczowe w języku Delphi możemy pisać zarówno dużymi, jak i małymi literami, natomiast w C++ Builderze tylko małymi literami.
3. Jeśli chcemy używać zmiennych w programie, to musimy wszystkie je wcześniej zadeklarować, informując kompilator o ich nazwie i typie.
4. Ogólna postać deklaracji zmiennej jest:

Delphi	*C++ Builder*

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

var **nazwa_zmiennej: typ zmiennej;**	**typ_zmiennej nazwa zmiennej;**

Typ zmiennej określa typ wartości (na przykład liczba całkowita lub zmiennoprzecinkowa), jaką może ona przechowywać, a także operacje, jakie program może na niej wykonywać.

5. Dla podniesienia czytelności programów, podczas deklarowania zmiennych należy używać nazw znaczących.
6. Ogólna postać instrukcji przypisania jest następująca:

Delphi	*C++ Builder*
nazwa_zmiennej := wyrażenie	**nazwa_zmiennej = wyrażenie**

Do wyświetlania na ekranie wartości zmiennej służy np. komponent **Edit**.

Copyright by Wydawnictwo Złote Myśli & Mirosław J. Kubiak

Jak skorzystać z wiedzy zawartej w pełnej wersji ebooka?

Więcej praktycznych porad dotyczących programowania w Delphi i C ++ Builder znajdziesz w pełnej wersji ebooka.

Zapoznaj się z opisem na stronie:

http://podstawy-delphi-builder.zlotemysli.pl

Jak szybko nauczyć się programowania w dwóch różnych językach?

POLECAMY TAKŻE PORADNIKI:

Sekrety języka C# (c-sharp) – Andrzej Stefańczyk

Dlaczego tworzenie aplikacji w Visual Studio .NET 2005 jest takie proste?

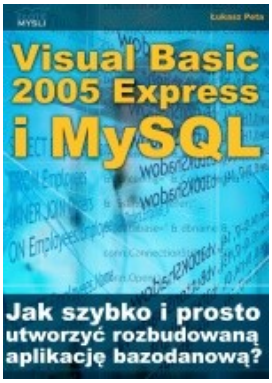
Ebook "Sekrety języka C#" uczy zarówno składni nowego języka C# jak również zasad tworzenia aplikacji okienkowych dla systemu Windows w tym języku.
Przeznaczony jest zarówno dla osób, które nie miały do tej pory styczności z programowaniem jak i dla osób, które programowały wcześniej w innych językach i chciałyby poznać możliwości C# (c-sharp).

Więcej o tym poradniku przeczytasz na stronie:
http://c-sharp.zlotemysli.pl

"Świetna pozycja dla chcących się nauczyć C# w .NET-cie zwięźle opisuje dany problem i przedstawia rozwiązania. Po prostu bomba."

- **webmax**, *student WISZ-u z Gorzowa Wlkp.*

Visual Basic 2005 Express i MySQL – Łukasz Peta



Jak szybko i prosto utworzyć rozbudowaną aplikacje bazodanową?

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

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

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

David 27 lat, programista

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