

Mirosław J. Kubiak

ZŁOTE
MYŚLI

Programuję w Delphi i C++ Builder

II część

dla zaawansowanych

**Jak szybko nauczyć
się programowania
w dwóch różnych językach**

Niniejszy ebook jest **własnością prywatną**.
Został zakupiony legalnie w serwisie Netpress.pl,
będącym oficjalnym Partnerem [Wydawcy](#).

Niniejsza publikacja, ani żadna jej część, nie może być kopiowana, ani w jakikolwiek inny sposób reprodukowana, powielana, ani odczytywana w środkach publicznego przekazu bez pisemnej zgody wydawcy. Zabrania się jej publicznego udostępniania w Internecie.

© Copyright for Polish edition by ZloteMysli.pl

Data: 11.10.2006

Tytuł: Programuję w Delphi i C++ Builder

Autor: Mirosław J. Kubiak

Wydanie I

ISBN: 83-7521-105-2

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 | 5 |
| ROZDZIAŁ 8. TABLICE | 7 |
| Deklarowanie tablic..... | 7 |
| Dostęp do elementów tablicy..... | 12 |
| Sito Eratostenesa..... | 15 |
| Tablice dwuwymiarowe..... | 18 |
| Sortowanie bąbelkowe..... | 24 |
| Tablice przechowują również teksty..... | 29 |
| Warto zapamiętać..... | 43 |
| ROZDZIAŁ 9. PODPROGRAMY | 44 |
| Co to są podprogramy..... | 44 |
| Definiujemy funkcje i procedury w języku Delphi..... | 44 |
| Definiujemy funkcje w języku C++ Builder..... | 49 |
| Programy mogą przekazywać podprogramom informacje..... | 53 |
| Przekazywanie parametrów przez wartość..... | 54 |
| Zmienne globalne i lokalne..... | 59 |
| Przekazujemy funkcjom i procedurom tablice..... | 60 |
| Rekurencja..... | 69 |
| Warto zapamiętać..... | 75 |
| ROZDZIAŁ 10. ELEMENTY GRAFIKI | 76 |
| Wprowadzenie | 76 |
| Program Przykładowa grafika..... | 85 |
| Graficzny przykład programu rekurencyjnego..... | 92 |
| Warto zapamiętać..... | 109 |
| ROZDZIAŁ 11. PRZECHOWYWANIE INFORMACJI W REKORDACH I W STRUKTURACH | 110 |
| Rekordy i struktury..... | 110 |
| Warto zapamiętać..... | 121 |
| ROZDZIAŁ 11. ELEMENTY PROGRAMOWANIA OBIEKTOWEGO | 122 |
| Wprowadzenie..... | 122 |
| Hermetyzacja danych..... | 144 |
| Dziedziczenie..... | 144 |
| Polimorfizm..... | 154 |
| Warto zapamiętać..... | 155 |
| ROZDZIAŁ 13. OPERACJE WEJŚCIA/WYJŚCIA – CZĘŚĆ II. PLIKI | 156 |
| Wprowadzenie..... | 156 |
| C++ Builder..... | 157 |
| Pliki tekstowe..... | 158 |
| Zapisywanie rekordów i struktur do pliku..... | 176 |
| C++ Builder..... | 186 |
| Warto zapamiętać..... | 194 |
| ROZDZIAŁ 14. WSKAŹNIKI | 196 |
| Wprowadzenie..... | 196 |
| Wskaźniki i tablice..... | 200 |
| Programy mogą przekazywać podprogramom informacje | 206 |

| | |
|---|------------|
| Przekazywanie parametrów przez wskaźnik..... | 206 |
| Przekazywanie parametrów przez referencję..... | 209 |
| Warto zapamiętać..... | 212 |
| ROZDZIAŁ 15. ZMIENNE DYNAMICZNE..... | 213 |
| Wprowadzenie..... | 213 |
| Zmienne dynamiczne do tablic..... | 218 |
| Klasy TStringList i TStringList..... | 221 |
| Warto zapamiętać..... | 223 |
| ROZDZIAŁ 16. ALGORYTMY NUMERYCZNE..... | 224 |
| Obliczanie sumy szeregu | 224 |
| Wyznaczenie miejsca zerowego funkcji metodą Newtona..... | 229 |
| Wyznaczanie miejsca zerowego funkcji | 235 |
| Obliczanie całki metodą prostokątów..... | 241 |
| DODATEK..... | 247 |
| D1. Formatowanie łańcuchów tekstowych..... | 247 |
| D2. Wybrane systemowe procedury konwersji typu..... | 249 |
| D3. Standardowe procedury obsługujące pliki (Delphi) | 251 |
| D4. Wyświetlanie komunikatów..... | 254 |
| D5. Wartości parametru Flags dotyczące liczby i rodzaju przycisków..... | 255 |
| D6. Grafika w Delphi i w C++ Builder – | 259 |
| BIBLIOGRAFIA..... | 263 |

Wstęp

Umiejętność pisania programów przynajmniej w dwóch językach programowania to wyzwanie dzisiejszych czasów adresowane do młodych programistów.

Przystępnie napisany podręcznik składa się z 16 rozdziałów i dodatku. Przedstawiony w nim materiał zawiera kurs równoczesnego programowania w językach Delphi Pascal oraz C++ Builder, bogato okraszony licznymi przykładami programów oraz wielu klasycznych algorytmów w obu językach z wykorzystaniem programowania wizualnego i zdarzeniowego, gdzie dodatkowo uwzględniono obsługę sytuacji wyjątkowych. Te nowatorskie propozycje mogą z powodzeniem zostać wykorzystane m.in. w indywidualnej nauce przez pasjonatów programowania.

Ze względów technicznych książka została podzielona na dwie części: część I zawiera rozdziały od 1 do 7, natomiast część II rozdziały od 8 do 16.

Rozdziały od 1 do 11 oraz od 13 do 16 zawierają kurs programowania strukturalnego w obu językach programowania, natomiast rozdział 12 zawiera elementy programowania obiektowego w Delphi i C++ Builder.

Część II właśnie masz przed sobą, a pierwsza część jest dostępna pod adresem: <http://podstawy-delphi-builder.zlotemysli.pl>

Rozdział 8. Tablice. W rozdziale dowiemy się w jaki sposób deklarujemy tablice jedno- i dwuwymiarowe, na czym polega sortowanie bąbelkowe oraz o tym, że tablice przechowują nie tylko liczby, ale również teksty.

Wstęp

Rozdział 9. Podprogramy. W rozdziale dowiemy się co to są podprogramy i do czego można je wykorzystać, jak definiujemy procedury i funkcje, jak przekazujemy informacje procedurom i funkcjom, co to są zmienne globalne i lokalne oraz co to jest rekurencja.

Rozdział 10. Elementy grafiki. W rozdziale nauczymy się pisać proste programy graficzne w Delphi i C++ Builder oraz przedstawimy graficzny przykład programu rekurencyjnego.

Rozdział 11. Przechowywanie informacji w rekordach i w strukturach. Rozdział zawiera informacje o rekordach i strukturach oraz w jaki sposób te informacje są przechowywane w języku Delphi i C++ Builder.

Rozdział 12. Elementy programowania obiektowego. W rozdziale poznamy elementy programowania obiektowego, nauczymy się pisać proste programy zawierające obiekty oraz dowiemy się na czym polega hermetyzacja danych, dziedziczenie i polimorfizm.

Rozdział 13. Operacje wejścia/wyjścia - część II. Pliki. W rozdziale opowiemy co to są pliki oraz dowiemy się jak zapisywać informacje do pliku i jak je odczytywać.

Rozdział 14. Wskaźniki. Ten rozdział przybliży nam zawiłości wskaźników oraz poznamy w nim sposoby przekazywania parametrów przez wskaźnik i przez referencję.

Rozdział 15. Zmienne dynamiczne. W rozdziale omówiono różnicę pomiędzy zmiennymi statycznymi a zmiennymi dynamicznymi oraz jak przydzielać i zwalniać pamięć dla zmiennych dynamicznych.

Rozdział 16. Algorytmy numeryczne. W rozdziale omówiono kilka wybranych algorytmów numerycznych.

Rozdział 8. Tablice

W tym rozdziale dowiemy się, w jaki sposób deklarujemy tablice jedno- i dwuwymiarowe, na czym polega sortowanie bąbelkowe oraz o tym, że tablice przechowują nie tylko liczby, ale również teksty.

Deklarowanie tablic

Tablica jest to struktura danych, która umożliwia przechowywanie w sposób zorganizowany wielu zmiennych tego samego typu (całkowitego, rzeczywistego itd.). Aby stworzyć taką strukturę musimy dokonać deklaracji tablicy. W deklaracji tablicy musimy określić typ wartości, jaki ma przechowywać tablica, a także liczbę jej elementów. Tablice mogą być jednowymiarowe, dwuwymiarowe itd.

Delphi

Oto ogólna postać deklarowania w języku Delphi tablicy jednowymiarowej i związanej z nią zmiennej.

type

identyfikator_tablicy = array[rozmiar_tablicy] of typ

var

nazwa_zmiennej : identyfikator_tablicy;

A oto przykład zadeklarowania tablicy jednowymiarowej o nazwie **dane** typu całkowitego, zawierającej 10 elementów.

type

tablica = array[1..10] of integer;

var

dane : tablica;

możliwy jest również inny poprawny zapis:

var

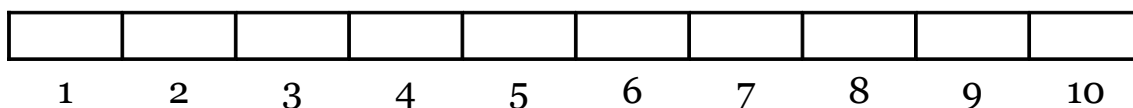
dane : array[1..10] of integer;

Dostęp do elementów tablicy jest realizowany za pośrednictwem indeksu, który wskazuje dany element. Dla deklaracji tablicy w języku Delphi zawartej poniżej:

var

dane : array[1..10] of integer;

pierwszy element tablicy **dane** ma indeks 1, drugi element dostępny jest przez indeks 2 itd. Ostatni element tablicy ma indeks równy wymiarowi tablicy, czyli 10, co zilustrowano na rysunku poniżej.



Oto prosty przykład programu ilustrujący posługiwanie się tablicą jednowymiarową.

Program (Tablica1)

Formularz

Na formularzu wykonujemy następujące czynności:

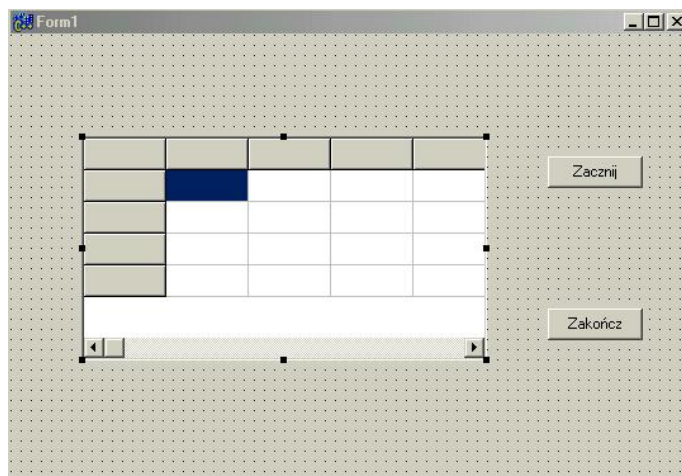
1. umieszczamy dwa przyciski **Button**, jeden u góry z prawej strony (**Button1**) i jeden na dole również z prawej strony (**Button2**),
2. następnie klikamy myszą raz na **Button1** i w *Object Inspectorze* zmieniamy we właściwościach (*Properties*) w *Caption* tekst **Button1** na **Zaczniij**,
3. klikamy myszą raz na **Button2** i w *Object Inspectorze* zmieniamy we właściwościach (*Properties*) w *Caption* tekst **Button2** na **Zakończ**,
4. z lewej górnej strony formularza umieszczamy komponent **StringGrid**,
5. dwukrotnie klikamy na formularzu przycisk **Zakończ** i podpinamy związaną z nim procedurę **procedure TForm1.Button2Click(Sender: TObject);**

begin

Close;

end;

6. następnie dwukrotnie klikamy na formularzu przycisk **Zaczniij** i podpinamy związaną z nim procedurę **procedure TForm1.Button1Click(Sender: TObject)**, która została opisana w programie poniżej.



Rys. 8.1. Formularz skonstruowany dla programu *Tablica1*, na którym zamieszczono komponent *StringGrid*.

```
unit Unit1;
```

```
interface
```

```
uses
```

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

```
type
```

```
TForm1 = class(TForm)
```

```
Button1: TButton;
```

```
Button2: TButton;
```

```
StringGrid1: TStringGrid;
```

```
procedure Button1Click(Sender: TObject);
```

```
procedure Button2Click(Sender: TObject);
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
Form1: TForm1;  
  
implementation  
  
{$R *.dfm}  
  
procedure TForm1.Button1Click(Sender: TObject);  
const  
  n = 10;  
  
type  
  tablica = array[1..n] of integer;  
var  
  i : integer;  
  dane : tablica;  
  
begin  
  StringGrid1.ColCount:=2; // ustalenie ilosci kolumn  
  StringGrid1.RowCount:=n+1; //ustalenie ilosci wierszy  
  StringGrid1.Cells[0,0]='Indeks tabl.';  
  StringGrid1.Cells[1,0]='Wart. tabl.';  
  
  for i:=1 to n do  
    StringGrid1.Cells[0,i]:=IntToStr(i); //opis wierszy  
  
  for i:=1 to n do  
    begin  
      dane[i]:=i;  
      StringGrid1.Cells[1,i]:=IntToStr(dane[i]);  
    end;  
  end;  
  
procedure TForm1.Button2Click(Sender: TObject);  
begin  
  Close();  
end;  
  
end.
```

C++ Builder

Oto ogólna postać zadeklarowania w języku w C++ Builder tablicy jednowymiarowej i związanej z nią zmiennej.

typ_tablicy nazwa_tablicy [rozmiar_tablicy]

A oto przykład zadeklarowania tablicy jednowymiarowej o nazwie **dane** typu całkowitego, zawierającej 10 elementów:

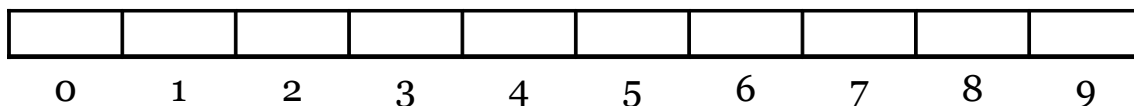
```
int dane [10];
```

Dostęp do elementów tablicy

Dostęp do elementów tablicy jest realizowany za pośrednictwem indeksu, który wskazuje dany element. Dla deklaracji tablicy

```
int dane [10];
```

aby uzyskać dostęp do pierwszego elementu tablicy **dane**, powinniśmy podać indeks 0¹, drugi element dostępny jest przez indeks 1 itd. Ostatni element tablicy ma indeks równy rozmiarowi tablicy minus 1, czyli 9, co zilustrowano na rysunku poniżej.



Oto prosty przykład ilustrujący posługiwanie się tablicą jednowymiarową.

¹Indeks pierwszego elementu tablicy w języku C++ wynosi zawsze 0.

Program (Tablica1)

Formularz

Na formularzu wykonujemy następujące czynności:

1. umieszczamy dwa przyciski **Button**, jeden u góry z prawej strony (**Button1**) i jeden na dole również z prawej strony (**Button2**),
2. następnie klikamy myszą raz na **Button1** i w *Object Inspectorze* zmieniamy we właściwościach (*Properties*) w *Caption* tekst **Button1** na **Zaczynij**,
3. klikamy myszą raz na **Button2** i w *Object Inspectorze* zmieniamy we właściwościach (*Properties*) w *Caption* tekst **Button2** na **Zakończ**,
4. z lewej górnej strony formularza umieszczamy komponent **StringGrid**,
5. dwukrotnie klikamy na formularzu przycisk **Zakończ** i podpinamy związaną z nim funkcję **void __fastcall TForm1::Button2Click(TObject *Sender)**

```
{  
  
    Close();  
  
}
```
6. następnie dwukrotnie klikamy na formularzu przycisk **Zaczynij** i podpinamy związaną z nim funkcję **void __fastcall TForm1::Button1Click(TObject *Sender)**, która została opisana w programie poniżej.

```
//-----  
#include <vcl.h>  
#pragma hdrstop  
  
#include "Unit1.h"  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TForm1 *Form1;  
//-----  

```

```
}  
//-----  
void __fastcall TForm1::Button2Click(TObject *Sender)  
{  
  Close();  
}  
//-----
```

Sito Eratostenesa

Okolo roku 200 p.n.e. matematyk grecki Eratostenes podał algorytm na znajdowanie liczb pierwszych. Liczby pierwsze to są liczby naturalne większe od 1, które dzielą się przez siebie i przez 1. Nazwa tego najstarszego algorytmu na znajdowanie liczb pierwszych pochodzi od sposobu, w jaki te liczby są znajdowane. Wszystkie liczby po kolei przesiewa się, usuwając z spośród nich wszystkie wielokrotności danej liczby. Zilustrujemy to przykładem, znajdując za pomocą sita Eratostenesa wszystkie liczby pierwsze z zakresu od 2 do 30, które umieścimy w tabeli poniżej.

| | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | |

Liczba 2 jest liczbą pierwszą, pozostałe znajdujemy po usunięciu z tabeli wielokrotności liczby 2 (gdyż nie są to liczby pierwsze) otrzymując:

| | | | | | | | | | | | | | | |
|----|---|----|---|----|---|----|---|----|----|----|----|----|----|---|
| 2 | 3 | * | 5 | * | 7 | * | 9 | * | 11 | * | 13 | * | 15 | * |
| 17 | * | 19 | * | 21 | * | 23 | * | 25 | * | 27 | * | 29 | * | |

Kolejny etap polega na usunięciu z tabeli po liczbie 3 (przyjmuje się, że jest to liczba pierwsza) wielokrotności liczby 3. Rezultat tego działania znajduje się poniżej:

| | | | | | | | | | | | | | | |
|----|---|----|---|---|---|----|---|----|----|---|----|----|---|---|
| 2 | 3 | * | 5 | * | 7 | * | * | * | 11 | * | 13 | * | * | * |
| 17 | * | 19 | * | * | * | 23 | * | 25 | * | * | * | 29 | * | |

Z pozostałych teraz liczb kolejną po 2 i 3 jest liczba 5, którą pozostawia się wykreślając wszystkie liczby podzielne przez 5:

| | | | | | | | | | | | | | | |
|----|---|----|---|---|---|----|---|---|----|---|----|----|---|---|
| 2 | 3 | * | 5 | * | 7 | * | * | * | 11 | * | 13 | * | * | * |
| 17 | * | 19 | * | * | * | 23 | * | * | * | * | * | 29 | * | |

Kontynuując to wykreślanie dojdziemy do sytuacji, kiedy zostaną wykreślone wszystkie liczby, które nie są pierwsze i pozostaną tylko liczby pierwsze.

W tym momencie możemy zakończyć nasze poszukiwania liczb pierwszych pamiętając, że kolejne wykreślenia należy powtarzać, nie dalej jak do liczby będącej zaokrąglonym w dół pierwiastkiem kwadratowym ze zmiennej **zakres**. W naszym przykładzie jest to: $\text{sqrt}(30)=5.477\dots$, po zaokrągleniu w dół² otrzymujemy liczbę 5. W tabeli pozostały już tylko liczby pierwsze, które wyświetlamy na ekranie.

Górny zakres, dla którego chcemy odnaleźć liczby pierwsze wyznaczony jest tylko rozmiarem tablicy. W naszym programie jest to:

```
int tablica[10000];
```

Chcąc zmienić ten zakres, należy zmienić rozmiar tablicy.

²W języku C++ Builder realizuje to matematyczna funkcja **floor()**, która znajduje się w pliku nagłówkowym **math.h**.

C++ Builder (Sito)

```
//-----  
#include <vcl.h>  
#include <math.h>  
#pragma hdrstop  
  
#include "Unit1.h"  
//-----  
#pragma package(smart_init)  
#pragma resource "*.dfm"  
TForm1 *Form1;  
//-----  

```

```
for (i=2; i<=zm_pom; i++)
  if (tablica[i]!=0)
    for (j=i+1; j<=zakres; j++)
      if (j%i==0) tablica[j]=0;

ListBox1->Items->Clear();
ListBox1->Items->Add("Liczby pierwsze z zakresu od 1 do
"+IntToStr(zakres)+" to");

for (i=2; i<=zakres; i++)
  if (tablica[i]!=0) ListBox1->Items->Add(IntToStr(i));
}
//-----
```