Mirosław J. Kubiak

Turbo Pascal

Zadania z programowania z przykładowymi rozwiązaniami

Turbo Pascal w analizie konkretnych przykładów

- *Proste operacje wejścia/wyjścia*
- *Tablice, iteracje oraz instrukcje warunkowe*
- *Programowanie obiektowe i pliki tekstowe*

Helion

Turbo Pascal. Zadania z programowania z przykładowymi rozwiązaniami

Autor: Mirosław Kubiak
ISBN: 978-83-246-2942-8
Format: 140×208, stron: 128

Turbo Pascal w analizie konkretnych przykładów

- Proste operacje wejścia/wyjścia
- Tablice, iteracje oraz instrukcje warunkowe
- Programowanie obiektowe i pliki tekstowe

Turbo Pascal to wciąż popularny proceduralny język programowania, który doskonale nadaje się do nauki programowania strukturalnego. Dobry programista, student lub nauczyciel informatyki, a także każdy człowiek zainteresowany programowaniem powinien znać podstawy tego języka i umieć rozwiązywać konkretne zadania. Podobnie zresztą powinien opanować najważniejsze zagadnienia dotyczące programowania w językach Java i C++ – i stosować je w praktyce. Trzyczęściowy zbiór, w którym zamieszczono te same lub bardzo zbliżone zadania wraz z rozwiązaniami w każdym z wyżej wymienionych języków, pozwala sprawdzić i uzupełnić wiedzę poprzez analizę podanego kodu.

Książka „Turbo Pascal. Zadania z programowania z przykładowymi rozwiązaniami" to jedna z trzech części zbioru zadań programistycznych, zawierająca zadania w języku Turbo Pascal. Dzięki ich analizie zrozumiesz, na czym polegają operacje wejścia/wyjścia, do czego można używać instrukcji warunkowych oraz iteracji, jak wykorzystać tablice jedno ¬ i dwuwymiarowe. Nauczysz się stosować elementy programowania obiektowego w tym języku oraz dowiesz się więcej o plikach tekstowych. Taki układ książki ułatwi Ci przyswojenie sobie najważniejszych zagadnień z Turbo Pascala w najlepszy możliwy sposób – na prostych, konkretnych przykładach.

- Proste operacje wejścia/wyjścia
- Instrukcje warunkowe i instrukcje wyboru
- Iteracje
- Tablice jedno- i dwuwymiarowe
- Programowanie obiektowe
- Rekurencja
- Pliki tekstowe

Praktycznie opanuj podstawy języka Turbo Pascal

Helion SA
ul. Kościuszki 1c
44-100 Gliwice
tel. 32 230 98 63
e-mail: helion@helion.pl

Spis treści

1

Proste operacje wejścia-wyjścia

W tym rozdziale zamieszczono proste zadania z przykładowymi rozwiązaniami ilustrujące, w jaki sposób komputer komunikuje się z użytkownikiem w języku Turbo Pascal.

Każda aplikacja powinna posiadać możliwość komunikowania się z użytkownikiem. Wykorzystując prosty przykład pokażemy, w jaki sposób program napisany w języku Turbo Pascal komunikuje się z nim poprzez standardowe operacje wejścia-wyjścia.

Instrukcje wyjścia

Do wyprowadzania danych na ekran służą dwie instrukcje (procedury[1] standardowe): `Writeln` i `Write`. Instrukcja `Writeln` powoduje wyprowadzenie danych na ekran monitora i automatyczne przejście kursora do nowej linii. Jej ogólna postać jest następująca:

```
Writeln(lista argumentów);
```

gdzie *lista argumentów* może być ciągiem znaków stałych, zmiennych lub wyrażeń oddzielonych od siebie przecinkami.

[1] Więcej informacji o procedurach znajdzie czytelnik w rozdziale 5.

Instrukcja `Write` umożliwia wyprowadzenie danych na ekran monitora, nie powodując automatycznego przejścia kursora do nowej linii. Jej ogólna postać to:

```
Write(lista argumentów);
```

gdzie *lista argumentów* również może być ciągiem znaków stałych, zmiennych lub wyrażeń oddzielonych za pomocą przecinków.

Instrukcje wyjścia `Writeln` i `Write` umożliwiają przedstawienie liczb w postaci sformatowanej, tj. z określoną liczbą miejsc przed i po kropce dziesiętnej. Aby uzyskać sformatowaną postać liczby rzeczywistej, należy argument tych funkcji uzupełnić o określenie szerokości pól w następującej postaci:

```
: szerokość pola: liczba miejsc po kropce
```

Zapis `Write(suma:6:2)` oznacza, że wartość zmiennej `suma` zostanie wyświetlona w polu o szerokości sześciu znaków z dwoma cyframi po kropce.

Instrukcje wejścia

Do wprowadzania zmiennych do uruchomionego programu w Turbo Pascalu służą dwie instrukcje (procedury standardowe): `Read` i `Readln`. Instrukcja `Read` umożliwia wprowadzenie do uruchomionego programu wartości zmiennych z klawiatury, nie powodując automatycznego przejścia kursora do nowej linii. Jej ogólna postać jest następująca:

```
Read(lista argumentów);
```

gdzie *lista argumentów* może być ciągiem znaków stałych, zmiennych lub wyrażeń oddzielonych od siebie przecinkami.

`Readln` również jest instrukcją umożliwiającą wprowadzenie do uruchomionego programu wartości zmiennych z klawiatury, powoduje ona jednak (po wprowadzeniu danych) automatyczne przejście kursora do nowej linii. Jej ogólna postać jest następująca:

```
Readln(lista argumentów);
```

gdzie *lista argumentów* może być ciągiem znaków stałych, zmiennych lub wyrażeń oddzielonych od siebie przecinkami.

ZADANIE

1.1 Napisz program, który oblicza pole prostokąta. Wartości boków `a` i `b` wprowadzamy z klawiatury. W programie należy przyjąć, że zmienne `a` i `b` oraz `pole` są typu `Real` (rzeczywistego). Dla zmiennych tych przyjmujemy format wyświetlania ich na ekranie w polu czteroznakowym z dwoma miejscami po kropce.

Przykładowe rozwiązanie — listing 1.1

```
program Project1; // Zadanie 1.1

{$APPTYPE CONSOLE}

uses
  SysUtils;

var
  a, b, pole: Real; // deklarujemy zmienne typu Real

begin
  Writeln('Program oblicza pole prostokata.');
  Writeln('Podaj bok a.');
  Readln(a);
  Writeln('Podaj bok b.');
  Readln(b);
  pole := a*b; // obliczamy pole prostokata
  Write('Pole prostokata o boku a = ', a:4:2, ' i boku b = ', b:4:2);
  Write(' wynosi ', pole:4:2, '.');

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Zmienne określonego typu deklarujemy w programie za pomocą słowa kluczowego `var`. Linijki kodu

```
var
  a, b, pole: Real; // deklarujemy zmienne typu Real
```

umożliwiają deklarację zmiennych `a`, `b` i `pole`. Wszystkie te zmienne są typu rzeczywistego — `Real`. Instrukcja

```
Writeln('Program oblicza pole prostokata.');
```

wyświetla na ekranie komputera komunikat *Program oblicza pole prostokata*. Instrukcja `Readln(a)` czeka na wprowadzenie z klawiatury liczby, która następnie zostanie przypisana zmiennej `a`. Pole prostokąta zostaje obliczone w instrukcji

```
pole := a*b; // obliczamy pole prostokata
```

Za wyświetlenie wartości zmiennych `a`, `b` oraz `pole` wraz z odpowiednim opisem odpowiedzialne są następujące linijki kodu:

```
Write('Pole prostokata o boku a = ', a:4:2, ' i boku b = ', b:4:2);
Write(' wynosi ', pole:4:2, '.');
```

Instrukcja

```
Readln; // czeka na nacisniecie klawisza Enter
```

czeka na użycie klawisza *Enter*, aby po jego naciśnięciu zamknąć ekran działającego programu. Komentarze oznaczamy w programie dwoma ukośnikami lub dwoma nawiasami klamrowymi:

```
// to jest komentarz do kodu
{to też jest komentarz do kodu}
```

Komentarze są ignorowane przez kompilator w procesie kompilacji.

Rezultat działania programu można zobaczyć na rysunku 1.1.

```
Program oblicza pole prostokata.
Podaj bok a.
1
Podaj bok b.
2
Pole prostokata o boku a = 1.00 i boku b = 2.00 wynosi 2.00.
```

Rysunek 1.1. *Efekt działania programu Zadanie 1.1*

ZADANIE

1.2

Napisz program, który wyświetla na ekranie komputera wartość predefiniowanej stałej $\pi = 3{,}14\ldots$ Należy przyjąć format wyświetlania jej w polu 10-znakowym z ośmioma miejscami po kropce.

Przykładowe rozwiązanie — listing 1.2

```
program Project1; // Zadanie 1.2

{$APPTYPE CONSOLE}

uses
  SysUtils;
```

```
begin
  Writeln('Program wyswietla wartosc predefiniowanej stalej pi');
  Writeln('z dokladnoscia osmiu miejsc po kropce.');
  Writeln('Pi = ', pi:10:8);

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Rezultat działania programu można zobaczyć na rysunku 1.2.

```
Program wyswietla wartosc predefiniowanej stalej pi
z dokladnoscia osmiu miejsc po kropce.
Pi = 3.14159265
```

Rysunek 1.2. *Efekt działania programu Zadanie 1.2*

ZADANIE

1.3 Napisz program, który wyświetla na ekranie komputera pierwiastek kwadratowy z wartości predefiniowanej $\pi = 3{,}14\ldots$ Należy przyjąć format wyświetlania wyniku w polu 10-znakowym z ośmioma miejscami po kropce.

Przykładowe rozwiązanie — listing 1.3

```
program Project1; // Zadanie 1.3

{$APPTYPE CONSOLE}

uses
  SysUtils;

begin
  Writeln('Program wyswietla pierwiastek kwadratowy z wartosci
  ↪predefiniowanej pi');
  Writeln('z dokladnoscia osmiu miejsc po kropce.');
  Writeln('Sqrt(pi) = ', Sqrt(pi):10:8);

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Pierwiastek kwadratowy ze stałej pi obliczamy za pomocą funkcji `Sqrt()`.

Rezultat działania programu można zobaczyć na rysunku 1.3.

```
Program wyswietla pierwiastek kwadratowy z wartosci
predefiniowanej pi
z dokladnoscia osmiu miejsc po kropce.
Sqrt(pi) = 1.77245385
```

Rysunek 1.3. *Efekt działania programu Zadanie 1.3*

ZADANIE

1.4 Napisz program, który oblicza objętość kuli o promieniu `r`. Wartość promienia wprowadzamy z klawiatury. W programie należy przyjąć, że promień `r` jest typu `Real` (rzeczywistego). Dla zmiennych `r` oraz `objetosc` przyjmujemy format wyświetlania ich na ekranie w polu czteroznakowym z dwoma miejscami po kropce.

Przykładowe rozwiązanie — listing 1.4

```
program Project1; // Zadanie 1.4

{$APPTYPE CONSOLE}

uses
  SysUtils;

var
  r, objetosc: Real;

begin
  Writeln('Program oblicza objetosc kuli o promieniu r.');
  Writeln('Podaj promien r.');
  Readln(r);
  objetosc := 4*Pi*r*r*r/3;
  Write('Objetosc kuli o promieniu r = ', r:4:2);
  Writeln(' wynosi ', objetosc:4:2,'.');

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Objętość kuli oblicza następująca linijka kodu:

```
objetosc := 4*Pi*r*r*r/3;
```

gdzie potęgowanie zamieniono na mnożenie.

Rezultat działania programu można zobaczyć na rysunku 1.4.

```
Program oblicza objetosc kuli o promieniu r.
Podaj promien r.
1
Objetosc kuli o promieniu r = 1.00 wynosi 4.19.
```

Rysunek 1.4. *Efekt działania programu Zadanie 1.4*

ZADANIE

1.5 Napisz program obliczający wynik dzielenia całkowitego bez reszty dwóch liczb `a = 37` i `b = 11`.

> **Wskazówka**
>
> Należy zastosować operator dzielenia całkowitego bez reszty `div`. Umożliwia on uzyskanie całkowitej wartości liczbowej z wyniku dzielenia, podczas gdy reszta jest odrzucana.

Przykładowe rozwiązanie — listing 1.5

```
program Project1; // Zadanie 1.5

{$APPTYPE CONSOLE}

uses
  SysUtils;

const
  a = 37;
  b = 11;

begin
  Writeln('Program oblicza wynik dzielenia calkowitego bez reszty dwoch
  ↪liczb.');
  Writeln; // wyswietlenie pustej linii
  Writeln('Dla liczb a = ', a, ' i b = ', b);
  Writeln(a, ' div ', b, ' = ', a div b, '.');

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Stałe `a` i `b` definiujemy w programie za pomocą słowa kluczowego `const`, tak jak pokazują następujące linijki kodu:

```
const
   a = 37;
   b = 11;
```

Rezultat działania programu można zobaczyć na rysunku 1.5.

```
Program oblicza wynik dzielenia calkowitego bez reszty
dwoch liczb.

Dla liczb a = 37 i b = 11
37 div 11 = 3.
```

Rysunek 1.5. *Efekt działania programu Zadanie 1.5*

ZADANIE

1.6 Napisz program, który oblicza resztę z całkowitego dzielenia dwóch liczb `a = 37` i `b = 11`.

> **Wskazówka**
>
> Należy zastosować operator reszty z całkowitego dzielenia `mod` (modulo). Umożliwia on uzyskanie tylko reszty z dzielenia, natomiast całkowita wartość liczbowa jest odrzucana.

Przykładowe rozwiązanie — listing 1.6

```
program Project1; // Zadanie 1.6

{$APPTYPE CONSOLE}

uses
  SysUtils;

const
  a = 37;
  b = 11;

begin
  Writeln('Program oblicza reszte z calkowitego dzielenia dwoch liczb.');
  Writeln;
  Writeln('Dla liczb a = ', a, ' i b = ', b);
  Writeln(a, ' mod ', b, ' = ', a mod b, '.');

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Rezultat działania programu można zobaczyć na rysunku 1.6.

```
Program oblicza reszte z calkowitego dzielenia dwoch liczb.

Dla liczb a = 37 i b = 11
37 mod 11 = 4.
```

Rysunek 1.6. *Efekt działania programu Zadanie 1.6*

ZADANIE

1.7 Napisz program, który oblicza sumę, różnicę, iloczyn i iloraz dwóch liczb `x` i `y` wprowadzanych z klawiatury. W programie należy przyjąć, że liczby `x` i `y` są typu `Real` (rzeczywistego). Dla zmiennych `x`, `y`, `suma`, `roznica`, `iloczyn` i `iloraz` przyjmujemy format wyświetlania ich na ekranie w polu czteroznakowym z dwoma miejscami po kropce.

Przykładowe rozwiązanie — listing 1.7

```
program Project1; // Zadanie 1.7

{$APPTYPE CONSOLE}

uses
  SysUtils;

var
  x, y : Real;
  suma, roznica, iloczyn, iloraz : Real;

begin
  Writeln('Program oblicza sume, roznice, iloczyn i iloraz dwoch liczb.');
  Writeln('Podaj x.');
  Readln(x);
  Writeln('Podaj y.');
  Readln(y);

  suma := x+y;
  roznica := x-y;
  iloczyn := x*y;
  iloraz := x/y;

  Writeln('Dla x = ', x:4:2, ' i y = ', y:4:2);
  Writeln; // wyswietlenie pustej linii
  Writeln('suma = ',suma:4:2, ',');
  Writeln('roznica = ', roznica:4:2, ',');
```

```
  Writeln('iloczyn = ', iloczyn:4:2, ',');
  Writeln('iloraz = ', iloraz:4:2, '.');

  Readln; // czeka na nacisniecie klawisza Enter
end.
```

Za obliczenie w programie sumy, różnicy, iloczynu i ilorazu odpowiadają następujące linijki kodu:

```
suma := x+y;
roznica := x-y;
iloczyn := x*y;
iloraz := x/y;
```

Rezultat działania programu można zobaczyć na rysunku 1.7.

```
Program oblicza sume, roznice, iloczyn i iloraz dwoch liczb.
Podaj x.
3
Podaj y.
2
Dla x = 3.00 i y = 2.00

suma = 5.00,
roznica = 1.00,
iloczyn = 6.00,
iloraz = 1.50.
```

Rysunek 1.7. *Efekt działania programu Zadanie 1.7*

» Turbo Pascal to wciąż popularny proceduralny język programowania, który doskonale nadaje się do nauki programowania strukturalnego. Dobry programista, student lub nauczyciel informatyki, a także każdy człowiek zainteresowany programowaniem powinien znać podstawy tego języka i umieć rozwiązywać konkretne zadania. Podobnie zresztą powinien opanować najważniejsze zagadnienia dotyczące programowania w językach Java i C++ — i stosować je w praktyce. Trzyczęściowy zbiór, w którym zamieszczono te same lub bardzo zbliżone zadania wraz z rozwiązaniami w każdym z wyżej wymienionych języków, pozwala sprawdzić i uzupełnić wiedzę poprzez analizę podanego kodu.

» Książka *Turbo Pascal. Zadania z programowania z przykładowymi rozwiązaniami* to jedna z trzech części zbioru zadań programistycznych, zawierająca zadania w języku Turbo Pascal. Dzięki ich analizie zrozumiesz, na czym polegają operacje wejścia/wyjścia, do czego można używać instrukcji warunkowych oraz iteracji, jak wykorzystać tablice jedno- i dwuwymiarowe. Nauczysz się stosować elementy programowania obiektowego w tym języku oraz dowiesz się więcej o plikach tekstowych. Taki układ książki ułatwi Ci przyswojenie sobie najważniejszych zagadnień z Turbo Pascala w najlepszy możliwy sposób — na prostych, konkretnych przykładach.

- *Proste operacje wejścia/wyjścia*
- *Instrukcje warunkowe i instrukcje wyboru*
- *Iteracje*
- *Tablice jedno- i dwuwymiarowe*
- *Programowanie obiektowe*
- *Rekurencja*
- *Pliki tekstowe*

Praktycznie opanuj podstawy języka Turbo Pascal.

Nr katalogowy: 5802

Księgarnia internetowa:
http://helion.pl

Zamówienia telefoniczne:
0 801 339900
0 601 339900

Helion

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

helion.pl
helion.pl
księgarnia internetowa

Cena: 19,90 zł

ISBN 978-83-246-2942-8