Pascal - powtórka

Alfabet, Nazwy

W odróżnieniu do C w Pascal nie odróżnia małych i dużych liter. Zapisy ALA i ala oznaczają tę samą nazwę. Podobnie np. słowo kluczowe **for** może być zapisane: **FOR**.

W Pascalu identyfikatorem zmiennej może być tylko taki ciąg znaków, który zaczyna się od litery, a składa się z liter, cyfr i znaku podkreślenia.

Typy proste

Całkowite:

```
typ                 zakres              sposob zapamietania
shortint
integer          -32768    32767         2 bajty (z bitem znaku)
longint
byte
word
```

Rzeczywiste

```
typ                zakres               liczba cyfr      rozmiar
                   38         -37       znaczacych       w bajtach
real         2.9*10   ..1.7*10            11 - 12            6
single
double

extended
comp
```

Funkcje standardowe o wartościach całkowitych:

```
 a)  o argumencie całkowitym:
       abs(i)       moduł 1i1
       sqr(i)       kwadrat i
       succ(i)      następnik i+1
       pred(i)      poprzednik i-1
 b)  o argumencie rzeczywistym:
    round(r)     najbliższa liczba całkowita (zaokrąglenie)
              round(4,8) = 5
              round(-4,8)=-5
    trunc(r)     najbliższa  całkowita w kierunku zera (obcięcie)
              trunc(4,8) = 4
              trunc(-4,8)=-4

Funkcje standardowe o wartościach rzeczywistych:
      abs(x)      1x1
      sqr(x)      x
      sqrt(x)      rx
      ln(x)       lnx
      exp(x)      e
      sin(x)      sinx
      cos(x)      cosx
      arctan(x)   arc tgx
      Frac(x)     cześć ułamkowa argumentu
```

```
Int(x)      cześć całkowita argumentu
pi          zwraca wartość pi=3.141592653589793238
```

Na wartościach całkowitych mogą być wykonane następujące operacje:

```
+     dodawanie
-     odejmowanie
*     mnożenie
div   dzielenie całkowite
mod   branie reszty z dzielenia całkowitego
```

Operacje na wartościach rzeczywistych:

```
+  dodawanie
-  odejmowanie
*  mnożenie
/  dzielenie
```

Typ znakowy

Zbiorem wartości tego typu (mającego identyfikator CHAR) jest zbiór znaków, uporządkowany zgodnie z rozszerzonym zbiorem znaków ASCII.
Stała znakową jest każdy napis znakowy o długości 1.
Np.: ‘a’ ‘A’ ‘-‘ ‘8’ ‘$’

Typ logiczny

Używany identyfikator typu: BOOLEAN. Tylko dwie wartości: FALSE i TRUE.

Wartości wszystkich typów można porównywać operatorami relacji:

< = > <= >= <>

Oba argumenty muszą być typów zgodnych. Dopuszczalne jest porównywanie całkowitych z rzeczywistymi.
Wynik relacji jest typu **boolean**.

W wyrażeniach mogą występować elementy różnych typów, jednak poprawne wyrażenia to takie, w których każdy z operatorów działa na argumentach o właściwym typie (np. oba argumenty operatora div są całkowite).
Typem wyrażenia nazywamy typ wartości wynikowej wyrażenia.
Kolejność wykonywania działań wskazują nawiasy, a poza nimi priorytet operatorów - tj. ustalone reguły pierwszeństwa.
Priorytet operatora zależy od klasy, do której należy dany operator.
Poznane dotychczas operatory nalezą do dwóch klas:
- operatory multiplikatywne: * div mod /
- operatory addytywne: + -

Operatory multiplikatywne maja pierwszeństwo przed addytywnymi.
Operacje o tym samym priorytecie realizowane są w kolejności zapisu (od lewej do prawej).

Struktura programu:

Program nazwa; { nagłówek można opuścić }

{ deklaracje w skład których mogą wchodzić sekcje deklaracji:
- stałych rozpoczynające się słowem: CONST
- zmiennych VAR
- typów TYPE
- procedury PROCEDURE
- funkcji FUNCTION }

begin

{ instrukcje }

end.

Przykład

```
Program przykład;
Const dziesiec = 10;                    { używany znak = }

Type calkowite = Integer;               { używany znak = }
Var   a, b : real;                      { używany znak : }
      i : integer;
      k: Calkowite;
      znak : char;
Begin
Read(b, i);
a := B;
K:=I*(dziesiec div 10);
ZNAK:='=';
Writeln('Liczba a', Znak, a, ' Liczba k ', znaK, k);
End.
```

W części deklaracyjnej mogą pojawić się wyłącznie deklaracje. W części wykonawczej (pomiędzy słowami **begin** i **end**) tylko instrukcje.

Porównanie instrukcji:

Przypisanie

Pascalu stosowany jest dwuznak: :=.

Zmienna **:=** Wyrażenie

Typy Zmiennej i Wyrażenia muszą być zgodne (w sensie przypisania). Dopuszczalne jest przypisanie wartości całkowitej pod zmienną rzeczywistą. Dokonywana jest konwersja.

Czytanie

Read(lista zmiennych);
Readln;
Readln(lista zmiennych);

Instrukcje Readln wymagają potwierdzenia danych klawiszem Enter.

Można wczytywać dane typów całkowitych, rzeczywistych, **char** i string.

Typ napisowy string

Służy do przechowywania wartości łańcuchowych (w standardzie do 255 znaków).

Np.

```
Var     nap : string;           { długość domyślna 255 znaków }
        Lan : string[10];       { ograniczenie długości łańcucha do 10 znaków }
        Znak : char

. . . . .

nap := 'Ala ma kota';           { Użycie stałej napisowej }
readln(lan);
znak := nap[1];
writeln('Litera A=', znak, ' Litera m=', nap[5],' Napis = ', nap);
```

Wyprowadzanie

Write(lista wyrażeń);
Writeln(lista wyrażeń);

Instrukcja Writeln kończy wyprowadzanie znakiem nowej linii.
Lista wyrażeń może zawierać wyrażenia typów prostych i string.

C	**Pascal**
Instrukcja grupująca	
{ <*zestaw instrukcji*> }	**begin** <*zestaw instrukcji*> **end**
Instrukcje warunkowe	
if (*wyrażenie*) *instrukcja1*;	**if** *wyrażenie* **then** *instrukcja1*;
lub	
if (*wyrażenie*) *instrukcja1*; **else** *instrukcja2*;	**if** *wyrażenie* **then** *instrukcja1* **else** *instrukcja2*
switch (*wyrażenie_całkowite*) **{** **case** *wartość_1* : *instrukcja_1*; **break**;; **case** *wartość_2* : *instrukcja_2*; **break**;; **. . .** **default** : *instrukcja*; **break**; **}**	**case** *wyrażenie_proste* **of** *wartość_1* : *instrukcja_1*; *wartość_2* : *instrukcja_2*; **. . .** **else** instrukcja **end**;
Instrukcje pętli	
do *instrukcja* **while** (*wyrażenie*)	**repeat** *instrukcje* **until** *warunek_końca*;
while (*wyrażenie*) *instrukcja*	**while** *wyrażenie* **do** *instrukcja*
for (*inst_inicjujace* ; *wyr_test* ; *inst_krok*) *instr_petli* ;	**for** zmienna := wyr_pocz **to** wyr_kon **do** instrukcja; lub: **for** zmienna := wyr_pocz **downto** wyr_kon **do** instrukcja;

Oba języki używają skoków: **goto**, **break**, **continue i exit** o znaczeniu i składni podobnych.

Przykład użycia:

```
var i: integer;
begin
for i:=1 to 10 do
  begin
if (i>2) and (i<9) then continue;
write(i,' ');                          {1  2  9  10}
  end;
```

```
var i: integer;
begin
for i:=1 to 10 do
  begin
  write(i,' ');
  if (i>2) and (i<9) then break;        {1  2}
  end;
```

Definiowane typy proste w pascalu.

Typ okrojony:

[*wartosc_dolna* **..** *wartosc_gorna* **]**

Typ taki obejmuje wartości określone zakresem. Oba ograniczenia muszą być jednakowego typu prostego nie rzeczywistego.

Typ wyliczeniowy

(*stała1*, *stała2*, ... **)**

Stałe muszą być identyfikatorami. Typ zawiera tyle wartości ile jest wymienione w deklaracji. Deklaracja ta służy jednocześnie do zdefiniowania stałych.

Deklaracja tablic

Obowiązująca składnia opisu typu tablicowego:

Array [*typ_prosty1*, ... **] of** *typ_elementu*;

Tablica może mieć wiele wymiarów.

Przykłady

```
Type
        Tab1 = array [1 .. 10] of char;
        Tab2 = array[ 1.. 10, 1..10 ] of integer;
Var
        Nap: array [ 1..20 ] of Tab1;
        X : array [ 1 .. 10 ] of real;
        Y1, Y2 : Tab2;

Type
        Kolor: (czarne, czerwone, niebieskie, zielone, biale )
Var
        Indeks: Kolor;
        I: Integer;
        Tab: array[Kolor] of integer;
        RGB: array [czerwone .. zielone ] of integer;
```

Wykorzystanie np:

```
i:=0;
for Indeks:= czarne to biale do
  begin
  Inc(i);
  Tab[Indeks]:=i;
  end;
  writen(Tab[czarne]):        {1}
  writen(Tab[Inc(czarne)]):   {2}
  writen(Tab[Dec(biale)]):    {4}
```

Deklaracja rekordów

Rekord jest odpowiednikiem struktury w języku C.

```
record
nazwa_pola1 : typ1;
nazwa_pola2 : typ2;
 . . .
end;
```

Procedury i funkcje

W pascalu funkcje nie przekazujące wyniku (w języku C typu void) nazywa się procedurami.

Składnia funkcji i procedury

function *nazwa*(*lista_parametrów*) : *typ_wyniku*;
deklaracje_lokalne
begin
instrukcje
nazwa := *wyrażenie*;
end;

procedure *nazwa*(*lista parametrów*);
deklaracje_lokalne
begin
instrukcje
end;

Parametry podprogramów mogą być przekazywane:

- przez wartość,
- przez zmienną.

Zabiegi wykonywane dodatkowo w momencie przekazywania parametrów z segmentu

wywołującego do podprogramu są inne w przypadku przekazywania przez wartość i przez zmienna.
Przekazywanie przez wartość powoduje wykonanie dodatkowych niejawnych instrukcji przypisań typu:

parametr formalny := parametr aktualny.

Przekazywanie przez zmienną powoduje skutki równoznaczne z zastąpieniem w tekście instrukcji złożonej procedury nazw parametrów formalnych odpowiednimi nazwami parametrów aktualnych.

Deklaracja parametru przez zmienną jest poprzedzona słowem **var**. Taka deklaracja jest stosowana dla parametrów przenoszących wyniki obliczone w procedurze.

Przykład

```
procedure oblicz(x: real; var y: real);
begin
x:=x+1;
y:=x;
end;
var a, b: real;
begin
a:=5;
oblicz(a, b);
writeln(a, b);
end.
```

Program powyższy wyświetli liczby: 5 i 6. Ponieważ parametr **x** jest wiązany przez wartość to jego zmiany wewnątrz procedury nie przenoszą się do programu głównego. Inaczej parametr **y** wiązany przez zmienną.