Elementi programskog jezika Pascal

Osnovni elementi jezika

Osnovni simboli

U programskom jeziku Pascal sve konstrukcije se grade od skupa osnovnih simbola jezika koji čine slova, cifre i specijalni znaci.

Slova

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z

Cifre

0	1	2	3	4	5	6	7	8	9

Specijalni znaci

+	-	*	/	=	<	>	[	]	.	,	;	:	^	(	)	’	{	}

Pored navedenih znakova u specijalne znake spadaju i znaci koji se ne štampaju, kao što su praznina, znak za novi red, tabulator itd.

Rezervisane reči

Osnovni elementi jezika su i **rezervisane reči** koje imaju unapred definisano značenje i ne mogu se koristiti u druge svrhe.

Rezervisane reči

and	array	begin	case	const	div	do
downto	else	end	file	for	function	goto
if	in	label	mod	nil	not	of
or	packed	procedure	program	record	repeat	set
then	to	type	until	var	while	with

Rezervisane reči označavaju različite elemente jezika čije će značenje biti detaljnije objašnjeno.

Većina Pascal kompajlera ne pravi razliku između velikih i malih slova, tako da je svejedno da li ćete napisati **begin** ili **BEgiN**. Na samom programeru je da odluči kojim stilom će pisati program, ali u svakom slučaju treba da vodi računa o tome da program bude čitljiv i lako razumljiv i za druge članove tima.

Tipovi podataka

Podacima se naziva sve ono što može biti predmet obrade računarom ili se može dobiti kao rezultat obrade. To mogu biti brojevi, slova, skupovi, slogovi i dr. složeni oblici.

U računaru su svi tipovi podataka predstavljeni u binarnom obliku, odnosno određenim brojem nula i jedinica (bitova). Radi lakše manipulacije podacima, u višim programskim jezicima postoje različiti tipovi podataka. Kompajleri prepoznaju određene tipove podataka i prevode ih u binarni oblik po određenim pravilima. Takođe, prilikom čitanja podataka iz memorije kompajleri prevode podatke iz binarnog oblika u odgovarajuće tipove podataka viših programskih jezika.

Tip podataka određuje skup vrednosti koje podatak može imati. U Pascal-u se tipovi podataka mogu podeliti u tri grupe:

1. Prosti tipovi

2. Složeni tipovi
3. Pokazivači

U ovom poglavlju navode se samo prosti tipovi podataka, a o složenim tipovima i pokazivačima će biti reči kasnije.

Celobrojni tip – Integer

Za rad sa celim brojevima koristi se **celobrojni tip** podataka. Ovaj tip podatka se u programskom jeziku Pascal označava rezervisanom rečju **Integer**.

Primeri celobrojnih vrednosti mogu biti:

28	728	6495	6
-56	+34	-9	-4825

Znak + ispred celobrojne vrednosti nije obavezan. Ukoliko ispred celobrojne vrednosti ne postoji predznak, smatra se da je vrednost pozitivna.

Vrednost najmanjeg i najvećeg celog broja zavisi od sistema na kome se radi, tako da na jednom tipu računara raspon celih brojeva može biti od -32768 do 32767, dok na drugom može biti od -2147483648 do 2147483647.

Da li možda znate...

Zašto je minimalni ceo broj po apsolutnoj vrednosti za jedan veći od maksimalnog celog broja?

Realan tip – Real

Brojevi koji pored celobrojnog sadrže i decimalni deo nazivaju se realnim brojevima. U programskom jeziku Pascal ovi brojevi se označavaju rezervisanom rečju **Real**. Prilikom pisanja realnih brojeva, celobrojni i decimalni deo se razdvajaju **decimalnom tačkom**[1].

Primeri realnih vrednosti mogu biti:

7.6	-4.62	7.584	-3000.0
-565.24	+34.81	-9.0	0.085

Pored predstavljenog načina pisanja realnih brojeva, programski jezik Pascal omogućava i pisanje realnih brojeva sa **pokretnom tačkom**[2] (floating point). Ovakav način pisanja je naročito prikladan za vrlo velike i vrlo male brojeve. Tako na primer broj 32 100 000, može biti zapisan kao 32.1E+6, što je ekvivalentno zapisu $32.1 \cdot 10^6$. Sa druge strane, broj 0.000658 može biti zapisan kao 6.58E-4, što je isto kao i $6.58 \cdot 10^{-4}$.

Ovakvo pisanje realnih brojeva se naziva još i eksponencijalna notacija. Eksponent (broj naveden iza slova "E") označava za koliko mesta treba pomeriti decimalnu tačku. Pozitivan eksponent označava da decimalnu tačku treba pomeriti u desno, dok negativan znak pomera decimalnu tačku u levo. Drugačije gledano, eksponent predstavlja stepen broja 10 kojim treba pomnožiti navedeni realan broj.

Ukoliko je eksponent pozitivan broj, predznak + iza slova E je moguće izostaviti. Takođe, ukoliko realni broj ne sadrži decimalni deo i on se može izostaviti.

Evo nekih primera pisanja realnih brojeva sa pokretnim zarezom:

Eksponencijalni zapis (pokretna tačka)	Standardni zapis
6.3E+4	63000
2.526E2	252.6
4.72E-2	0.0472
782.8E-1	78.28
45E+2	4500
0.25E-5	0.0000025

[1] Pascal za razdvajanje celobrojnog i decimalnog dela ne koristi decimalni zarez, već decimalnu tačku.

[2] Iako je u srpskom jeziku uobičajen termin pokretni zarez, s obzirom da Pascal koristi decimalnu tačku, u ovom tekstu ćemo koristiti termin **pokretna tačka**

Neke bitne napomene

Brojevi 342 i 342.0 u matematici imaju istu vrednost, međutim u radu na računaru oni često označavaju različite tipove podataka. Tako, u nekim Pascal kompajlerima 342 predstavlja celobrojni tip, dok će 342.0 biti protumačen kao realni tip podatka. Da biste uvek bili sigurni da je broj koji ste napisali pravilno prtumačen bez obzira na kompajler koji koristite, najbolja praksa je da broj za koji želite da bude tretiran kao realan uvek pišete sa decimalnom tačkom, bez obzira da li on ima ili nema decimalnih cifara.

Neki Pascal kompajleri dozvoljavaju skraćeno pisanje realnih brojeva koji nemaju decimalnih cifara tako da se oni završavaju decimalnom tačkom. Na taj način, iako broj nema decimalnih cifara, kompajleru se stavlja do znanja da je u pitanju realan, a ne celobrojni tip. Tako se broj "342" može zapisati kao "342.", što je ekvivalentno zapisu "342.0". Takođe, ukoliko realan broj nema celobrojnog dela (po apsolutnoj vrednosti je manji od 1), u nekim kompajlerima on se može zapisati bez cifara levo od decimalne tačke. Na primer, broj "0.12" se može zapisati i kao ".12". Iako veliki broj kompajlera podržava ovakve načine zapisivanja realnih brojeva, preporuka je da se uvek koristi neskraćeno zapisivanje kako bi napisani program bio kompatibilan sa svim kompajlerima.

Evo nekih primera pomenutih načina zapisivanja realnih brojeva:

Skraćeni zapis	Potpuni zapis
23.	23.0
.756	0.756
-.5	-0.5
6.E+3	6.0E+3
.48E-3	0.48E-3

Prilikom rada sa realnim brojevima uvek treba imati u vidu da tačnost zavisi od broja cifara pomoću kojih je broj zapisan. Broj značajnih cifara, a samim tim i preciznost računanja, zavisi od broja bajtova koji se koriste za zapisivanje realnih brojeva. Pored toga, operacije nad realnim brojevima u zavisnosti od preciznosti troše više procesorskog vremena od operacija nad celim brojevima.

Logički tip – Boolean

Logički tip podataka se koristi za označavanje istinitosti nekog iskaza i može imati dve vrednosti:

Vrednost	Značenje
false	netačno
true	tačno

U programskom jeziku Pascal se logički tip podatka označava rezervisanom rečju **Boolean**. Vrlo često se logičke vrednosti nazivaju i Boolovim vrednostima u čast engleskog matemaričara George Boole-a, koji je prvi razvio logičku algebru. U računaru se logičke vrednosti predstavljaju jednim bitom koji može imati vrednost 0 za netačno i 1 za tačno.

Znakovni tip – Char

Znakovni tip podataka je uređen skup znakova koji mogu biti:

- slova abecede
- numerički znakovi od 0 do 9
- znakovi interpunkcija
- specijalni znakovi

U programskom jeziku Pascal znakovni tip se označava rezervisanom rečju **Char**. Svakom znaku u skupu pridružen je jedan ceo broj koji se naziva kod. Najčešće korišćena tabela kodova je ASCII (**vidi prilog**).

U programskom jeziku Pascal znakovne vrednosti se zapisuju kao znak između jednostrukih navodnika:

'A'	'b'	'X'	'4'	'+'	'-'	'>'	'@'

Korišćenje navodnika pri pisanju znakova je neophodno kako bi se oni razlikovali od brojeva, operatora, promenljivih, itd. Na primer, 4 je celobrojna vrednost, a '4' znakovna vrednost.

Nizovi znakova kao što su reči i rečenice se u Pascal-u nazivaju stringovima i označavaju se rezervisanom rečju **String**.

Evo nekih primera stringova:

'Programiranje'	'Goran igra fudbal, a Jana vozi bicikl.'	'345.08'	'2+5=7'	'peraperic@yahoo.com'

Konstante

Odeljak za definisanje konstanti počinje rezervisanom reči **const** (od eng. reči *constant=konstanta*) iza koje se navode nazivi konstanti i njihovih vrednosti. Konstante i njihove vrednosti se razdvajaju znakom **=**, a definicije pojedinih konstanti znakom **;**. Vrednost konstante u Pascal-u može biti broj (ceo ili realan), string ili prethodno definisana konstanta. Pojedini kompajleri omogućavaju da vrednost konstante bude izraz koji se sastoji od drugih konstanti. Pri definisanju konstante ne navodi se njen tip, pošto je on jednoznačno određen vrednošću koja joj se dodeljuje. Vrednost definisane konstante nije moguće menjati u programu.

Sintaksa

```
const   < naziv_1 = vrednost_1; >
        [ naziv_2 = vrednost_2; ]
                  ...
        [ naziv_n = vrednost_n; ]
```

Primer

```
const   PI=3.14;
        email=’pera@yahoo.com’;
        g=9.81;
        metar=1;
        kilometar=1000*metar;
```

Korišćenje konstanti je veoma korisno u slučajevima kada se neka vrednost pojavljuje često u programu. U takvim slučajevima je iz praktičnih razloga pogodnije definisati konstantu sa datom vrednošću, a zatim svuda u programu umesto konkretne vrednosti koristiti definisanu konstantu. Na taj način omogućena je veoma jednostavna promena navedene vrednosti, tako što bi se promena izvršila samo u definiciji konstante, bez potrebe za promenama u ostatku programa.

Pisanje konkretnih vrednosti u programu je loša praksa. Umesto konkretnih vrednosti koristite prethodno definisanu konstantu. Na taj način omogućavate lakšu promenu vrednosti na jednom jedinom mestu, bez opasnosti da ćete propustiti da vrednost promenite u svim delovima programa. Takođe, dobro imenovana konstanta znatno poboljšava čitljivost koda.

Promenljive

Promenljive ili varijable su objekti čija vrednost može biti promenjena tokom izvršavanja programa. Svaka promenljiva koja se koristi u programu mora biti prethodno deklarisana. Deklaracijom promenljive ne određuje se njena vrednost, već samo tip podatka koji će u njoj biti smešten. Na osnovu navedenog tipa podatka svakoj promenljivoj se dodeljuje memoriski prostor odgovarajuće veličine u kome će biti upisana vrednost promenljive. Dodeljivanje odgovarajućeg memorijskog prostora (alokacija) se obavlja prilikom pokretanja programa ili potprograma, a njegova pozicija i veličina se ne mogu menjati tokom izvršenja, pa se iz tog razloga ovaj način alokacije naziva **statička alokacija**. Pored ovakvog načina dodeljivanja memorije postoji i **dinamička alokacija**, koji omogućava alokaciju memorije određene veličine u bilo kom trenutku tokom izvršavanja programa.

Odeljak za deklarisanje promenljivih počinje rezervisanom reči **var** (od eng. reči *variable=promenljiva*) iza koje se navode nazivi promenljivih i njihovi tipovi. Promenljive istog tipa mogu biti odvojene zapetom, iza čega sledi dvotačka i tip podatka.

Prilikom deklarisanja promenljivih tipa **string** moguće je u uglastim zagradama iza rezervisane reči **string** definisati i maksimalnu dužinu stringa. Ukoliko prilikom deklarisanja nije naveden ovaj podatak, podrazumeva se da je maksimalna dužina stringa 255.

Sintaksa

```
var  < naziv_11 [, naziv_12 [,naziv_13 [, ...]]] :tip_podatka_1; >
     [ naziv_21 [ naziv_22 [,naziv_23 [, ...]]] :tip_podatka_2; ]
                                  ...
     [ naziv_n1 [ naziv_n2 [,naziv_n3 [, ...]]] :tip_podatka_n; ]
```

Primer

```
var  x,y,z:real;
     ime:string;
     adresa:string[30];
     i,j:integer;
     pritisak:real;
     iskaz1,iskaz2:boolean;
     slovo:char;
```

Svaka promenljiva u programu predstavlja određenu veličinu ili pojam, pa im zbog toga treba uvek davati smislena imena. Osim određenog broja pomoćnih promenljivih, koje radi jednostavnosti možete označiti i samo jednim slovom (na primer, brojači u petljama su često *i, j*...), ostalim promenljivim treba davati nazive koji asociraju na to šta one zapravo predstavljaju (na primer: pritisak, ime, ocena...). Naravno, treba imati meru i u dužini naziva promenljivih, kako bi se izbegao suviše glomazan i nepregledan kod.
Programeri veoma često svoju duhovitost iskazuju tako što promenljivama daju neobična imena (*meshalica*, *kupusijada* i sl.), koja nemaju nikakve veze sa njihovim pravim značenjem. Ovakvi nazivi mogu na kratko biti interesantni, ali vrlo brzo ni sam programer nije siguran šta one zapravo znače, a da ne govorimo o ostalim članovima tima.
Nazovite stvari pravim imenima. Dobri nazivi promenljivih znatno olakšavaju pisanje i razumevanje programa.

Izrazi

Izrazi su kombinacije simbola koje definišu poredak i način izračunavanja neke vrednosti korišćenjem:

- operanda (konstante, promenljive, funkcije)
- operatora
- oblih zagrada

Operatori definišu koje je operacije potrebno izvriti nad operandima, a zagrade definišu redosled vršenja tih operacija. U slučajevima kada redosled vršenja operacija nije određen zagradama, primenjuju se pravila o prioritetu operacija:

Operatori	Prioritet
not	1 (najveći prioritet)
and * / div mod	2
or + -	3
in = > < >= <= <>	4 (najmanji prioritet)

Operatori

Nad svim tipovima podataka moguće je izvršiti određene operacije. Operacija preslikava konačan skup podataka (operande) u konačan skup podataka (rezultate). **Operatori** definišu operacije koje je potrebno izvršiti nad operandima da bi se dobio rezultat.

Operacije nad celobrojnim (integer) tipom podataka

Nad celobrojnim operandima se mogu izvoditi sledeće operacije koje daju celobrojni rezultat:

Operator	Operacija
*	množenje
div	celobrojno deljenje
mod	ostatak pri celobrojnom deljenju
+	sabiranje
-	oduzimanje

Operacije *****, **div** i **mod** imaju viši prioritet od operacija **+** i **-**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Primeri

```
3 * 7 = 21
15 div 2 = 7
15 mod 2 = 1
4 + 9 = 13
4 – 9 = -5
-4 + 9 = 5
3 * 7 div 5 mod 3 = 1
3 + 2 * 4 – 2 * 5 = 1
```

Operacije nad realnim (real) tipom podataka

Nad realnim operandima se mogu izvoditi sledeće operacije koje daju realan rezultat:

Operator	Operacija
*	množenje
/	deljenje
+	sabiranje
-	oduzimanje

Operacije ***** i **/** imaju viši prioritet od operacija **+** i **-**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Pri radu sa realnim brojevima treba voditi računa o tome da se oni u memoriji ne beleže apsolutno tačno, već sa određenom tačnošću (određenim brojem decimalnih mesta). Iz tog razloga su i rezultati operacija nad realnim brojevima približni.

Primeri

```
3.47 * 7.21 = 25.0187
8.0 / 2.0 = 4.0
1.0 / 3.0 = 0.3333333
1.0 / 3.0 * 3.0 = 0.9999999
2.71 + 6.525 = 9.235
-7.812 – 0.1 = -7.912
6.17 + 2.14 * 3.81 - 4.5 = 9.8234
```

Operacije nad logičkim (boolean) tipom podataka

Nad logičkim operandima se mogu izvoditi sledeće operacije koje daju logički rezultat:

Operator	Operacija
not	negacija
and	konjukcija
or	disjunkcija
xor	eksluzivna disjunkcija

Operacija **not** ima viši prioritet od operacija **and** i **or**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Primeri

p	q	not p	p and q	p or q	p xor q
false	false	true	false	false	false
true	false	false	false	true	true
false	true	true	false	true	true
true	true	false	true	true	false

Relacijski operatori

Relacijski operatori omogućavaju poređenje dva operanda istog tipa, a dobijeni rezultat je logičkog tipa. Iako su celi i realni brojevi formalno različiti tipovi podataka, korišćenjem relacijskih operatora moguće ih je međusobno porediti.

Relacijski operatori

Operator	Relacija
=	jednako
<>	različito
>	veće
<	manje
>=	veće ili jednako
<=	manje ili jednako
Operatori nad logičkim tipom	
=	ekvivalencija
<>	ekskluzivna disjunkcija
<=	implikacija
Operatori nad skupovima	
=	jednakost skupova
<>	različitost skupova
<=	podskup
>=	nadskup
in	element skupa

Operatori **=** i **<>** mogu se primeniti i na promenljive tipa **array** i **record** ukoliko su one istog tipa.

Relacijski operatori su najnižeg prioriteta i obavljaju se tek pošto se prethodno obave svi aritmetički operatori.

U slučaju primene navedenih operatora na znakovni tip podatka, rezultat se dobije poređenjem njihovih ASCII kodova, tj. njihovih pozicija u ASCII tabeli. Tako je rezultat relacije 'A'<'C' jednak **true**, pošto je ASCII kod znaka 'A' 65, a znaka 'C' 67.

Primeri

Izraz	Vrednost
`5 <= 0`	`false`
`(2*2) <> (3*3)`	`true`
`true < false`	`false`
`(5 > 0) > (5 < 0)`	`true`
`(5 > 0) = (5 <> 0)`	`true`
`(5 < 7) and (7 < 9) <= (9 < 5)`	`false`
`'C' < 'M'`	`true`

Standardne funkcije

Pored operacija za koje su definisani operatori, postoje i operacije koje se mogu izvršiti korišćenjem standardnih funkcija. Standardne funkcije imaju samo jedan argument čiji je tip unapred definisan, a rezultat funkcije je takođe unapred definisanog tipa. Argument funkcije može biti i izraz odgovarajućeg tipa. Funkcije se u izrazima kotiste na sličan način kao i operatori, sa tom razlikom što se, umesto znaka koji predstavlja operaciju, navodi naziv funkcije, a zatim odgovarajući argument u zagradama.

U sledećoj tabeli navedene su standardne funkcije, njihovo značenje i odgovarajući tip argumenta i rezultata:

Funkcija	Operacija	Tip argumenta	Tip rezultata
`abs(x)`	apsolutna vrednost od **x**	integer	integer
`sqr(x)`	kvadrat od **x**	integer	integer
`succ(x)`	sledbenik od **x**, tj. **x+1**	integer	integer
`pred(x)`	prethodnik od **x**, tj. **x-1**	integer	integer
`sin(x)`	sinus od **x**	integer	real
`cos(x)`	kosinus od **x**	integer	real
`arctan(x)`	arkus tangens od **x**	integer	real
`ln(x)`	prirodni logaritam od **x**	integer	real
`exp(x)`	stepen **x** za osnovu **e**	integer	real
`sqrt(x)`	kvadratni koren iz **x**	integer	real
`odd(x)`	da li je **x** neparan broj	integer	boolean
`abs(x)`	apsolutna vrednost od **x**	real	real
`sqr(x)`	kvadrat od **x**	real	real
`sin(x)`	sinus od **x**	real	real
`cos(x)`	kosinus od **x**	real	real
`arctan(x)`	arkus tangens od **x**	real	real
`ln(x)`	prirodni logaritam od **x**	real	real
`exp(x)`	stepen **x** za osnovu **e**	real	real
`sqrt(x)`	kvadratni koren iz **x**	real	real
`trunc(x)`	ceo deo realnog broja **x**	real	integer
`round(x)`	zaokruživanje broja **x** na najbliži ceo broj	real	integer
`frac(x)`	decimalni deo realnog broja **x**	real	real
`ord(x)`	redni broj (kod) znaka **x** u ASCII tabeli	char	integer
`chr(x)`	znak čiji je redni broj (kod) u ASCII tabeli **x**	integer	char

Operator dodele

Operator dodele je operator koji određenoj promenljivoj (levi operand) dodeljuje vrednost izraza sa desne strane (desni operand). Dodeljivanje se vrši tako što se prvo izračuna vrednost izraza sa desne strane operatora, a zatim se ta vrednost upisuje u memorijsku lokaciju promenljive sa leve strane operatora. Zahvaljujući tome, ukoliko izraz u sebi sadrži i promenljivu koja je sa leve strane operatora, prvo će biti sračunata vrednost izraza koristeći staru vrednost promenljive, a tek onda će vrednost izraza biti dodeljena promenljivoj sa leve strane.

Tip vrednosti izraza na desnoj strani mora odgovarati tipu promenljive kojoj se vrednost dodeljuje. Jedini izuzetak je dodeljivanje celobrojnog izraza realnoj promenljivoj. U tom slučaju dobijena vrednost izraza se pretvara u realan broj i kao takva se dodeljuje realnoj promenljivoj sa leve strane.

Primeri

```
a:= 7;
i:= i+1;
suma:= suma+broj;
radijani:= stepeni*PI/180.0;
dobar:= (ocena>=2.5) and (ocena<3.5);
ime:= 'Pera';
```

Naredbe

Naredbe ili komande su elementi programa koji nalažu računaru da izvrši određenu akciju. Naredbe su međusobno razdvojene znakom **;**. Naredbe se u Pascalu mogu podeliti na **neizvršne** i **izvršne**. U neizvršne naredbe spadaju naredbe za definisanje i deklarisanje svega što će biti korišćeno u programu. Izvršne naredbe čine "radni" deo programa i u njih spadaju aritmetičko-logičke, upravljačke i ulazno izlazne naredbe.

Blokovi naredbi

Blokovi omogućavaju organizovanje naredbi u funkcionalne celine. Svaki blok naredbi počinje rezervisanom reči **begin**, a završava se rezervisanom reči **end**. Sve naredbe u okviru jednog bloka čine celinu i ne mogu se izvršavati odvojeno. Iz tog razloga se svaki blok može posmatrati kao jedna složena komanda.

Sintaksa

```
begin
 [komanda_1;]
 [komanda_2;]
   ...
 [komanda_n;]
end;
```

Primer

```
begin
   a:= 7;
   b:= a+1;
end;
```

Dobra praksa prilikom pisanja programa je korišćenje takozvane **nazubljene strukture**. To podrazumeva uvlačenje pojedinih redova (uz pomoć praznih mesta ili tabulatora), kako bi se vizuelno predstavila hijerarhija u programu. Iz tog razloga je dobra praksa uvlačenje komandi unutar bloka naredbi, kako bi se vizualno predstavila njihova pripadnost bloku. Nazubljena struktura ne utiče na funkcionalnost programa, ali njena upotreba znatno olakšava pisanje i čitanje koda, a samim tim i razumevanje njegove strukture.

Labele

Svaka naredba u programu može biti označena celim brojem od 0 do 9999 koji predstvlja njenu labelu ili oznaku. Komandi koja se označava prethodi labela praćena dvotačkom. Svaka labela koja se koristi u programu mora prethodno biti deklarisana korišćenjem rezervisane reči **label**. Korišćenjem komande **goto** moguće je izazvati skok programa na označenu komandu.

Sintaksa

```
label< labela1 [, labela2 [,labela3 [, ...]]]; >
```

Primer

```
label  54, 0100, 99;
```

KORIŠĆENJE LABELA SE STROGO NE PREPORUČUJE !
Korišćenje komandi **label** i **goto** omogućava direktan skok programa na određenu komandu. Međutim, ovakav način pisanja programa je loša praksa, jer se iz tako napisanog programa ne može lako pretpostaviti tok njegovog izvršenja. Umesto toga treba koristiti komande koje jasno definišu strukturu programa, o kojima će kasnije biti reči.
Labele kao jedan od elemenata programskog jezika Pascal su ovde navedene samo iz razloga kompletnosti teksta, a njihovo korišćenje se strogo ne preporučuje, jer ozbiljno narušava strukturu programa i otežava praćenje njegovog izvršenja.

Komentari

Svi navedeni elementi programskog jezika Pascal jednoznačno definišu njegovu funkcionalnost. Tako napisan program je iz perspektive računara potpuno čitljiv i ne zahteva nikakva dodatna objašnjenja. Međutim, za programera je u cilju boljeg razumevanja same funckionalnosti veoma važno da pojedine delove programa može detaljnije opisati jezikom koji je pristupačniji čoveku..

Komentari u Pascal-u se pišu između vitičastih zagrada i namenjeni su čoveku, a ne računaru. Tokom prevođenja programa ovako navedeni komentari se ignorišu i ne ulaze u izvršni kod. Komentar se može napisati u bilo kom delu programa, ali se ne sme pisati unutar ključne reči.

Struktura Pascal programa

Program sadrži niz naredbi, koje se izvršavaju nad određenim podacima u cilju dobijanja traženog rezultata. Kako bi program bio što čitljiviji najbolja praksa je pisanje jedne komande u svakom redu. Komande koje pripadaju određenom bloku treba pisati uvučeno za jednu ili dve pozicije. Sličan način pisanja treba koristiti i za ostale podređene strukture, kako bi se jasno označila njihova pripadnost drugoj programskoj strukturi. Programe uvek treba razbijati na manje logicke segmente koji čine zasebnu celinu. Promenljivama, funkcijama i ostalim elementima programa treba uvek davati nazive koji označavaju njihovu namenu. Program u kome su stvari nazvane pravim imenima je znatno čitljiviji i često ne zahteva dodatne komentare u kodu.

Svaki program se sastoji od:

- zaglavlja
- deklaracija labela
- definicija konstanti
- definicija tipova
- deklaracija promenljivih
- definicija funkcija i procedura
- odeljak naredbi

Osim odeljak naredbi svi ostali odeljci se mogu izostaviti, ukoliko nisu neophodni u programu.

Zaglavlje programa se sastoji od rezervisane reči **program** praćene nazivom programa, iza čega sledi znak za kraj komande **;**.

Odeljak naredbi počinje rezervisanom reči **begin**, a završava se rezervisanom reči **end**, iza čega sledi tačka kao znak za kraj programa.