Elementi programskog jezika C

Imajući u vidu da je cilj ovog kursa upoznavanje sa osnovama programiranja, u ovoj skripti će biti prikazani samo najosnovniji elementi programskog jezika C, na nivou koji je univerzalan za većinu programskih jezika. Kako bi se čitaocima koji se prvi put susreću sa programiranjem olakšalo razumevanje osnovnih koncepata programiranja, pojedini elementi programskog jezika C će biti pojednostavljeno objašnjeni, bez zalaženja u detalje. Detaljno i suštinsko objašnjenje takvih pojmova će biti predmet drugih kurseva.

Osnovni elementi jezika

Osnovni simboli

U programskom jeziku C 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

auto	struct	break	if	else	switch	case	enum
register	typedef	extern	union	for	continue	void	return
do	while	default	sizeof	volatile	const	double	int
long	char	float	short	static	signed	unsigned	const

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

C kompajler pravi razliku između velikih i malih slova, tako da je nije svejedno da li ćete napisati **while** ili **wHiLe**.

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 C-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 – int

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

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?

Realani tipovi – float, double, long double

Brojevi koji pored celobrojnog sadrže i decimalni deo nazivaju se realnim brojevima. U programskom jeziku C ovi brojevi se označavaju rezervisanim rečima **float**, **double** ili **long double**. 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 C 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.

[1] C 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 C koristi decimalnu tačku, u ovom tekstu ćemo koristiti termin **pokretna tačka**

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

Neke bitne napomene

Brojevi 342 i 342.0 u matematici imaju istu vrednost, međutim u radu na računaru oni često označavaju različite tipove podataka. Da biste uvek bili sigurni da je broj koji ste napisali pravilno protumačen 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 C 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 – bool[1]

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

Vrednost	Značenje
0	Netačno (false)
$\neq$ 0	Tačno (true)

U nekim kompajlerima programskog jezika C se logički tip podatka može označiti rezervisanom rečju **bool**. 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.

[1] **bool** tip podatka nije standardni element C jezika, ali se u nekim kompajlerima može uključiti korišćenjem specijalnih biblioteka. Neki noviji kompajleri podržavaju **bool** tip kao standardni deo C jezika.

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 C 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**).

1 ┌	33 !	65 A	97 a	129 ▯	161 ¡	193 Á	225 á
2 ┐	34 "	66 B	98 b	130 ,	162 ¢	194 Â	226 â
3 └	35 #	67 C	99 c	131 ƒ	163 £	195 Ă	227 ã
4 ┘	36 $	68 D	100 d	132 „	164 ¤	196 Ä	228 ä
5 |	37 %	69 E	101 e	133 …	165 ¥	197 Å	229 å
6 ─	38 &	70 F	102 f	134 †	166 ¦	198 Æ	230 æ
7 •	39 '	71 G	103 g	135 ‡	167 §	199 Ç	231 ç
8 ◘	40 (	72 H	104 h	136 ˆ	168 ¨	200 È	232 è
9	41)	73 I	105 i	137 ‰	169 ©	201 É	233 é
10	42 *	74 J	106 j	138 Š	170 ª	202 Ê	234 ê
11 ♂	43 +	75 K	107 k	139 ‹	171 «	203 Ë	235 ë
12 ▫	44 ,	76 L	108 l	140 Œ	172 ¬	204 Ì	236 ì
13	45 -	77 M	109 m	141 ▯	173 -	205 Í	237 í
14 ♫	46 .	78 N	110 n	142 Ž	174 ®	206 Î	238 î
15 ☼	47 /	79 O	111 o	143 ▯	175 ¯	207 Ï	239 ï
16 ┼	48 0	80 P	112 p	144 ▯	176 °	208 Đ	240 ð
17 ◄	49 1	81 Q	113 q	145 '	177 ±	209 Ñ	241 ñ
18 ↕	50 2	82 R	114 r	146 '	178 ²	210 Ò	242 ò
19 ‼	51 3	83 S	115 s	147 "	179 ³	211 Ó	243 ó
20 ¶	52 4	84 T	116 t	148 "	180 ´	212 Ô	244 ô
21 ┴	53 5	85 U	117 u	149 •	181 µ	213 Ő	245 ő
22 ┬	54 6	86 V	118 v	150 –	182 ¶	214 Ö	246 ö
23 ┤	55 7	87 W	119 w	151 —	183 ·	215 ×	247 ÷
24 ↑	56 8	88 X	120 x	152 ˜	184 ¸	216 Ø	248 ø
25 ├	57 9	89 Y	121 y	153 ™	185 ¹	217 Ù	249 ù
26 →	58 :	90 Z	122 z	154 š	186 º	218 Ú	250 ú
27 ←	59 ;	91 [	123 {	155 ›	187 »	219 Û	251 û
28	60 <	92 \	124 |	156 œ	188 ¼	220 Ü	252 ü
29	61 =	93]	125 }	157 ▯	189 ½	221 Ý	253 ý
30	62 >	94 ^	126 ~	158 ž	190 ¾	222 Þ	254 þ
31	63 ?	95 _	127 ⌂	159 Ÿ	191 ¿	223 ß	255 ÿ
32	64 @	96 `	128 €	160	192 À	224 à	

U programskom jeziku C 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 C-u nazivaju stringovima i označavaju se: **char** naziv_promenljive**[** broj_karaktera **]**.

Evo nekih primera stringova:

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

Konstante

Definisanje konstanti počinje rezervisanom reči **const** (od eng. reči *constant=konstanta*) iza koje se navodi naziv konstante i njena vrednost. Konstanta i njena vrednost se razdvajaju znakom **=**.Vrednost konstante u C-u može biti broj (ceo ili realan), niz ili prethodno definisana konstanta. Pojedini kompajleri omogućavaju da vrednost konstante bude izraz koji se sastoji od drugih konstanti. Vrednost definisane konstante nije moguće menjati u programu.

Sintaksa

```
const tip_promenljive naziv_promenljive = vrednost_promenljive;
```

Primer

```
const double PI = 3.14;
```

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 mora da se deklariše i 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.

Promenljuva se deklariše tako što se prvo napiše tip promenljive, naziv po želji i njena vrednost. Promenljive istog tipa mogu biti odvojene zapetom.

Sintaksa

```
tip_promenljive naziv_promenljive_1;

tip_promenljive naziv_promenljive_2 = vrednost_promenljive_2;

tip_promenljive naziv_promenljive_3, naziv_promenljive_4, naziv_promenljive_5;

tip_promenljive naziv_promenljive_6 = vrednost_promenljive_6, naziv_promenljive_7;
```

Primer

```
double x;
int y = 5;
float z, b, f;
char q, c = 'a';
```

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
- oblik zagrada

Operatori definišu koje je operacije potrebno izvršiti 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	Asocijativnost
++ -- () [] . -> (type){list}	1 (najveći)	Sa leva na desno
++ -- + - * & sizeof ! ~ (type)	2	Sa desna na levo
* / %	3	Sa leva na desno
- +	4	
<< >>	5	
< <= > >=	6	
== !=	7	
&	8	
^	9	
|	10	
&&	11	
||	12	
?:	13	Sa desna na levo
= += -= *= /= %= <<= >>= &= ^= |=	14	
,	15	Sa leva na desno

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 (int) tipom podataka

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

Operator	Operacija
*	množenje
/	celobrojno deljenje
%	ostatak pri celobrojnom deljenju
+	sabiranje
-	oduzimanje
++	uvećava vrednost za jedan
--	smanjuje vrednost za jedan
>>	bitovsko pomeranje u desno
<<	bitovsko pomeranje u levo
~	bitovski komplement
&	bitovska konjukcija
|	bitovska disjunkcija
^	bitovsko XOR

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

Primeri

```
3 * 7 = 21
15 / 2 = 7
15 % 2 = 1
4 + 9 = 13
4 – 9 = -5
-4 + 9 = 5
3 * 7 / 5 % 3 = 1
3 + 2 * 4 – 2 * 5 = 1
5++ = 6
++5 = 6
5-- = 4
--5 = 4
16 >> 1 = 8
16 << 1 = 32
17 & 9 = 1
17 | 9 = 25
~16 = -17
16 ^ 4 = 20
```

Operacije nad realnim tipovima (float, double, long double) 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 (bool) tipom podataka

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

Operator	Operacija
!	negacija
&&	konjukcija
||	disjunkcija
~	bitovski komplement
&	bitovska konjukcija
|	bitovska disjunkcija
^	bitovsko XOR

Operacija **!** ima viši prioritet od operacija **&&** i **||**. Operacije istog prioriteta se izvršavaju sa leva na desno.

Primeri

p	q	! p	p && q	p || q	p ^ q
netačno	netačno	tačno	netačno	netačno	netačno
tačno	netačno	netačno	netačno	tačno	tačno
netačno	tačno	tačno	netačno	tačno	tačno
tačno	tačno	netačno	tačno	tačno	netačno

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

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 **tačno**, pošto je ASCII kod znaka 'A' 65, a znaka 'C' 67.

Primeri

Izraz	Vrednost
`5 <= 0`	`netačno`
`(2*2) != (3*3)`	`tačno`
`true < false`[1]	`netačno`
`(5 > 0) > (5 < 0)`	`tačno`
`(5 > 0) == (5 != 0)`	`tačno`
`(5 < 7) && (7 < 9) <= (9 < 5)`	`netačno`
`'C' < 'M'`	`tačno`

Matematičke funkcije

Pored operacija za koje su definisani operatori, postoje i operacije koje se mogu izvršiti korišćenjem standardnih matematičkih funkcija. Da bi smo koristili ove matematičke funkcije potrebno je da u program uključimo biblioteku *math.h*, tako što na početku programa navedemo **#include <math.h>**. Argument funkcije može biti promenljiva i/ili 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 matematičke funkcije, njihovo značenje i odgovarajući tip argumenta i rezultata:

Funkcija	Operacija	Tip argumenta	Tip rezultata
`acos(x)`	arkus kosinus za vrednost **x**	double	double
`asin(x)`	arkus sinus za vrednost **x**	double	double
`atan(x)`	arkus tangens za vrednost **x**	double	double
`atan2(y, x)`	arkus tangens u radijusu y/x jer prepoznaje kvadrant	double	double
`cos(x)`	kosinus od **x**	double	double
`cosh(x)`	hiperbolički kosinus od **x**	double	double
`sin(x)`	sinus od **x**	double	double
`sinh(x)`	hiperbolički sinus od **x**	double	double
`tanh(x)`	hiperbolički tankgens od **x**	double	double
`exp(x)`	vraća vrednost **e** podignutu na stepen **x**	double	double
`log(x)`	prirodni logaritam broja **x**	double	double
`log10(x)`	logaritam sa osnovom 10	double	double
`pow(x, y)`	vraća vrednost **x** podignutu na stepen **y**	double, double	double
`sqrt(x)`	koren od **x**	double	double
`ceil(x)`	zaokružuje broj **x** na veću ili jednaku vrednost	double	double
`fabs(x)`	apsolutna vrednost od **x**	double	double
`floor(x)`	zaokružuje broj **x** na manju ili jednaku vrednost	double	double
`fmod(x, y)`	ostatak pri deljenju broja **x** brojem **y**	double, double	double

[1] Vrednosti **true** i **false** nisu standardni elementi C jezika, ali se u nekim kompajlerima mogu uključiti korišćenjem specijalnih biblioteka. Neki noviji kompajleri podržavaju **true** i **false** kao standardni deo C jezika.

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) && (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 C-u 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 rezervisanim znakom **{**, a završava se rezervisanim znakom **}**. 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

```
{
 [komanda_1;]
 [komanda_2;]
  ...
 [komanda_n;]
};
```

Primer

```
{
   a = 7;
   b = a+1;
};
```

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

Svakoj naredbi u programu može biti dodeljen naziv koji predstvlja njenu labelu ili oznaku. Komandi koja se označava prethodi labela praćena dvotačkom. Korišćenjem komande **goto** moguće je izazvati skok programa na označenu komandu.

Sintaksa

```
naziv_labele : naredba
naziv_labele : blok_naredbi
```

Primer

```
PrimerLabela: x = 5;
PrimerLabela2: {x = 5; y = 6;}
goto PrimerLabela;
```

KORIŠĆENJE LABELA SE STROGO NE PREPORUČUJE !
Korišćenje komande **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 C 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 C 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. Za opis koristimo komentare.

Ako je komentar u C-u u jednoj liniji mora da počinje sa ***//***. Ako komentar sadrži više linija mora da počunje sa ***/**** i završava sa ****/***.

Komentari su namenjeni 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 C 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 logičke 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:

- deklaracija labela
- definicija konstanti
- definicija tipova
- deklaracija promenljivih
- definicija funkcija

Glavni program (glavna funkcija) počinje rezervisanom reči **main()**, a potom rezervisanim znakom **{**, a završava se rezervisanim znakom **}**.