Operatori i izrazi

Izraz = operatori + operanadi

- **operatori - definišu radnje koje se izvršavaju nad podacima dajući određeni rezultat**

- **operandi - konstante, promenljive i rezultati podizraza**

Operatori i izrazi

Operatori:

- ✓ **unarni**
  - ✓ **postfiksni**
  - ✓ **prefiksni**
- ✓ **binarni (infiksni)**
- ✓ **ternarni (uslovni izraz)**
- ✓ **n-arni (funkcije)**

Operatori i izrazi

Grupe operatora:

- ✓ **adresni**
- ✓ **aritmetički**
- ✓ **relacioni**
- ✓ **za rad sa bitovima**
- ✓ **logički**
- ✓ **za dodelu vrednosti**
- ✓ **operatorređanja ,**

Operatori i izrazi

- **Adresni – mehanizam pristupanja objektima u memoriji preko adrese (specifičnost C-a)**

- **Operator zarez – formiranje niza međusobno nezavisnih izraza (formalno jedan izraz)**

  a+b, b+3*c, c+x;

Operatori i izrazi

Grupisanje operanada uz operatore:

- ✓ **zagrade (mogu biti ugneždene)**
- ✓ **prioritet (16 nivoa)**
- ✓ **asocijativnost (smer grupisanja)**
- ✓ **na istom prioritetu, grupisanje svih operatora u istom smeru**

Aritmetički operatori

- **numerički operandi (i char!)**
- **numerički rezultati**
- **unarni i binarni operatori**
- **poredak izvršavanja za asocijativne i komutativne operatore (npr. +,*) nije garantovan, prevodilac može da preuredi izraz**

Aritmetički operatori

- **binarni operatori:**

```
*   proizvod   (13)
/   količnik   (13)
%   ostatak    (13)
+   zbir       (12)
-   razlika    (12)
```

- **asocijativnost** **L→D**

Aritmetički operatori

```
a+b-c          // (a+b)/c
a+b*c          // a+(b/c)
a/b*c          // (a/b)*c
7/4*3          // 3
7*3/4          // 5
7./4.*3.       // 5.25
7*3./4.        // 5.25
7%3            // 1
9%3            // 0
```

Aritmetički operatori

Unarni operatori:

```
-  promena predznaka   (14)
+  bez uticaja         (14)
-- dekrementiranje     (14 --a ili 15 a--)
++ inkrementiranje     (14 ++a ili 15 a++)
```

- **asocijativnost D→L** (14)
- **asocijativnost L→D** (15)

Aritmetički operatori: Unarni operatori

-- i ++

- **operandi samo promenljive (ne i rezultati drugih izraza, moraju biti *lvrednost*)**
- **postfiksni: i++ daje rezultat staro i, pa ga uvećava (postinkrement)**
- **prefiksni ++i uvećava, pa vraća kao rezultat novo i (preinkrement)**

Aritmetički operatori

```
++i             // i=i+1
i++             // i=i+1
j=i++           // j=i, i=i+1
j=++i           // i=i+1, j=i
```

- **primer <u>bočnih efekata</u> – rezultati koji se dobijaju <u>pored</u> vrednosti izraza**

Relacioni operatori

- **binarni operatori**

  `>,>=,<,<=` (10)

  `==, !=` (9)

- **operandi numeričkog tipa**
- **rezultat logičkog tipa, istinito/neistinito (formalno _Bool, i to 1=true, 0=false, mogu se koristiti u aritmetičkim izrazima)**
- **asocijativnost L→D**

Relacioni operatori

Primeri:

```
5>7                 // 0
10<=20              // 1
8==13>5             // 8==(13>5) -> 8==1 -> 0
14>5<3              // (14>5)<3  -> 1<3  -> 1
a<b<5               // (a<b)<5 -> 0<5 ili 1<5 -> 1
a+5>=c-1.0/e        // (a+5)>=(c-(1.0/e))
```

Logički operatori

- **operandi i rezultati <u>logičkog</u> tipa _Bool (istinito = 1/neistinito = 0) kod operanada bilo koja nenulta vrednost je logička istina**

- **binarni operatori**

```
&&   logičko I       (5)   L->D
||   logičko ILI     (4)   L->D
```

- **unarni operator**

```
!    negacija        (14)  D->L
```

Logički operatori

kod uzastopnih && ili ||

- **prvo se izračunava levi operand**
- **izračunavanje se prekida kada se utvrdi konačan rezultat**
- **mogu da izostanu bočni efekti**

Logički operatori

```
a && b || c && d  // (a && b)||(c && d)
a == b && c >= d  // (a == b)&&(c >= d)
a + 1 != b || !c  // ((a + 1)!= b)||(!c)
a < b && b < 5 // da li je b između a i 5
```

Operatori po bitovima

- **specifičnost C-a**
  (kao u simboličkom mašinskom jeziku)
- **operandi i rezultat su <u>celobrojni</u>**
- **operacije nad parovima bitova**

```
&   logičko I            (8)    L->D
|   logičko ILI          (6)    L->D
^   isključivo ILI       (7)    L->D
~   komplementiranje     (14)   D->L (unarni)
```

Operatori sa bitovima

- **pomeranje levog operanda za broj mesta ukazan desnim (mora biti >=0)**

`<<`	`ulevo`	`(11)`	`L->D`	množenje sa 2^n
`>>`	`udesno`	`(11)`	`L->D`	deljenje sa 2^n

- **logičko za neoznačene, aritmetičko za označene**

Operatori po bitovima

```
0x1234 & 0x5678          // 0x1230
0x1234 | 0x5678          // 0x567c
0x1234 ^ 0x5678          // 0x444c
~0x1234                  // 0xedcb
!0x1234                  // 0
022222 &  055555         // 000000
022222 && 055555         // 1
00001 << 5               // 000040
0x3801 << 4              // 0x8010
0xff56 >> 4              // 0xfff5
0xff56u >> 4             // 0x0ff5
```

Uslovni izraz

- **ternarni operator (operandi - tri izraza)**
- **`uslov` je tipa _Bool**

```
uslov?izraz1:izraz2     (3)    D->L
```

Uslovni izraz

`uslov?izraz1:izraz2`

- **ako je `uslov` istinit (!=0) rezultat je vrednost `izraz1`, inače je `izraz2`**
- **izračunava se <u>samo jedan</u> od izraza (`izraz1` ili `izraz2`)**
- **ako su `izraz1` i `izraz2` <u>različitog</u> tipa => konverzija tipa u složeniji**

Uslovni izraz

Primeri:

- **maksimum:**
  `z = (a>b)?a:b`
- **konverzija u veliko slovo**
  `c=(c>='a'&& c<='z')?(c-'a'+'A'):c`
- **funkcija sign (znak)**
  `sign=(x>0)?1:(x<0)?-1:0`

Dodela vrednosti

- **operator**
  (a ne naredba kao u većini jezika!)
- **može više dodela u jednoj naredbi**
- **prioritet - 2**
- **asocijativnost D->L**

```
izraz1 = izraz2
izraz1 op= izraz2
  (izraz1 = izraz1 op izraz2)
```

Dodela vrednosti

izraz1 = izraz2

- **izraz1 – skalarna promenljiva ili komponenta niza (indeksiranje je isto operator)**
- **izraz2 – proizvoljan numerički izraz**

izraz1 op= izraz2

- **op: + - * / % << >> & | ^**

Dodela vrednosti

- **ako su različitog tipa, konverzija u tip izraz1**
- **bolje odgovara načinu na koji programer razmišlja**
- **konciznost (važna pogotovo za komplikovane indeksirane promenljive)**

Dodela vrednosti

- **Primer:**

```
y=a*x+b          // y=((a*x)+b)
d*=e+f           // d=(d*(e+f))
d=d*e+f          // d=((d*e)+f)
a=b=c=d+5        // a=(b=(c=(d+5)))
a=b++ +3*(c=d-3)
                 // a=((b++)+(3*(c=(d-3))))
a=b++ +3*c=d-3     greška
                 // a=(((b++)+(3*c))=(d-3))
```

Konverzija tipa

- **C dozvoljava <u>različite tipove</u> u jednom izrazu**

- **pretvara <u>prostiji</u> tip u <u>složeniji</u>, izračunava izraz i dodeli rezultatu <u>složeniji</u> tip**

Konverzija tipa

Označeni tipovi

- **signed char**
- **short**
- **int**
- **long**
- **long long**
- **float**
- **double**
- **long double**

Neoznačeni tipovi

- **_Byte**
- **unsigned char**
- **unsigned short**
- **unsigned int**
- **unsigned long**
- **unsigned long long**

Konverzija tipa

- **Prikazana hijerarhija je od užih ka širim tipovima**
- **Konverzija iz užeg u širi tip je proširujuća konverzija (daje ispravan rezultat; moguć gubitak tačnosti (npr iz long u float)**
- **Konverzija iz šireg u uži tip obično uzrokuje gubitak tačnosti, a često i besmislen rezultat**

Konverzija tipa

- **kod dodele vrednosti uvek se pretvara u tip odredišnog operanda**
- **kod dva operanda različitog tipa, uvek se pretvara u širi tip**
- **operandi tipa užeg od int (odnosno unsigned int) najpre se pretvaraju u tip int (odnosno unsigned int)**
- **duži float u kraći – zaokruživanjem mantise**
- **kraći int u duži – proširuje znak za označene, ubacuje nule za neoznačene**
- **char u int – zavisi od implementacije (da li se char smatra označenim)**
- **za int operande iste dužine neoznačena varijanta je složeniji tip od označene**

Konverzija tipa

eksplicitno pretvaranje u željeni tip (cast – prefiksni unarni operator)

```
(ime_tipa)     (14)    D->L
```

Konverzija tipa

```
5+6.              //5.+6. -> 11.
5/4*3.            //(5/4)*3. -> 1*3. -> 3.
3.*5/4            //(3.*5)/4 -> 15./4 -> 3.75
-1L < 1U          // 1
-1L < 1UL         // 0 !!!
(double) a
(unsigned int) a+b
                  // ((unsigned int)a)+b
```

Konverzija tipa

```
-1L < 1U
   0xffffffffL < 0x0001U
   0xffffffffL < 0x00000001L -> 1

-1L < 1UL
   0xffffffffL  < 0x00000001UL
   0xffffffffUL < 0x00000001UL -> 0
```

Lančanje izraza

više izraza čini jedan izraz
(izraz1, izraz2, ..., izrazN)

```
operator zarez     (1)    L->D
```

Lančanje izraza

- **prvo levi, pa desni**
- **najniži prioritet**
  (nisu potrebne zagrade)
- **<u>vrednost</u> i <u>tip</u> izraza**
  određuje izrazN (poslednji u lancu)

Lančanje izraza

```
a+5, b+=13, c++
```
(nezavisni)

```
X=132, y*=x+3, x--, z -= x*3 - --y
```
(zavisni)

Veličina podataka

- **sizeof**
  - ✓ **prefiksni unarni**
  - ✓ **prioritet 14, D->L**

- **operand – izraz ili identifikator tipa**
- **vraća veličinu alociranog memorijskog prostora u bajtovima**
- **izraz se ne izračunava**

Veličina podataka

tip rezultata size_t

(neki celobrojni iz <stddef.h>)

```
sizeof(char)    //  1
sizeof(niz)     //  veličina čitavog niza
                    (N*jedna komponenta)
```

Veličina podataka

Po standardu nije postavljen odnos za cele i realne, već je garantovano samo:

1<=sizeof(char) <= sizeof(short) <=sizeof(int) <= sizeof(long)<= sizeof(long long)

sizeof(float) <=sizeof(double) <= sizeof(long double)

Veličina podataka

Primeri:

```
sizeof(unsigned long int)
sizeof sizeof(char)
  == sizeof sizeof (long double)
sizeof x
sizeof (enum dani) == sizeof dan1
sizeof (Duzina)
```

Prioritet i poredak izračunavanja operatora

- **Standard uglavnom ne nalaže poredak izračunavanja operanada kod operatora**
- **npr. a = b*c + d/e**
- **zavisno od prevodioca radi se prvo b*c ili d/e**
- **poredak izračunavanja argumenata funkcije proizvoljan**
- **omogućava optimizacije**

Prioritet i poredak izračunavanja operatora

izuzeci

- **logički operatori && i ||**
  (prvo levi izraz)
- **operator zarez**
  (prvo levi izraz)
- **uslovni izraz**
  (prvo izraz1, pa onda
  samo jedan od preostala dva)

Prioritet i poredak izračunavanja operatora

- **loša programerska praksa**
  ako rezultat zavisi
  od poretka izračunavanja

- **bitno je za operatore**
  sa bočnim efektima
  (++, --, dodela vrednosti, funkcije)

Prioritet i poredak izračunavanja operatora

Primeri dvoznačnosti:

- **a=b*c+b++**
  - ✓ **u=b*c, v=b, b++, a=u+v**
  - ✓ **v=b, b++, u=b*c, a=u+v**
- **a=b*c+(b=d/e)**
  - ✓ **u=b*c, v=(b=d/e), a=u+v**
  - ✓ **v=(b=d/e), u=b*c, a=u+v**

Tabela operatora

pri.	#op	operator	asoc.
16	2	[] () . ->	->
15	1	a++ a– (tip) {}	->
14	1	! ~ ++a -- a + - * & (tip) sizeof _Alignof	<-
13	2	* / %	->
12	2	+ -	->
11	2	<< >>	->

Tabela operatora

pri.	#op	operator	asoc.
10	2	< <= > >=	->
9	2	== !=	->
8	2	&	->
7	2	^	->
6	2	|	->

Tabela operatora

pri.	#op	operator	asoc.
5	2	&&	->
4	2	||	->
3	3	?:	<-
2	2	= += -= *= /= %= &= ^= |= <<= >>=	<-
1	2	,	->

Upravljačke strukture

<u>Naredbe</u> – osnovne jedinice obrade

- **<u>proste</u> (izraz;)**
  specijalni slučaj – prazna naredba

- **<u>složene</u> (upravljačke strukture)**
  - ✓ **sekvenca**
  - ✓ **selekcije**
  - ✓ **ciklusi**
  - ✓ **skokovi**

Sekvenca

- **niz naredbi koje se izvršavaju po poretku u kojem su napisane**
- **ograničavači – {}**
- **slobodan format pisanja, preporuke:**
  - ✓ **uvlačenje blokova**
  - ✓ **zagrade u posebnim redovima**
  - ✓ **obično - jedna naredba, jedan red**

Sekvenca

ima status bloka:

- **na početku svake sekvence mogu stajati definicije promenljivih**
- **promenljive su lokalne za blok**
- **oblast važenja – od mesta definicije do kraja bloka**
- **ugnežđavanje blokova – pravila vidljivosti**

Sekvenca

razlika sekvence i niza izraza:

- **niz izraza je isto izraz,**
  vraća vrednost, može biti operand
  t=a, a=b, b=t;

- **sekvenca – niz naredbi**
  {int t=a; a=b; b=t;}

Selekcije

- **uslovno izvršavanje naredbi**
- **biraju jednu ili više naredbi (prostih ili složenih)**

if – else

izbor jedne od dve naredbe

if (uslov) naredba1 [else naredba2]

- **uslov je tipa _Bool**
- **uslov ≠ 0 ⇒ naredba1**
- **uslov = 0 ⇒ naredba2**

bez else – uslovni preskok

načini pisanja

Primeri if – else

```
if (a>b) {t=a; a=b; b=t;}

if (mesec == 12)
{
   mesec=1;
   ++godina;
}
else
   ++mesec;
```

if – else

- **naredbe u if-u mogu biti i složene upravljačke naredbe**
- **odsustvo završne reči – problem kod ugneždenih if sa jednim else**
- **else tada pripada najbližem ranije neuparenom if-u**

if – else

```
if (uslov1)
   if (uslov2)
      naredba1;
   else
      naredba2;

if (uslov1) {
   if (uslov2)
      naredba1;
   } else
      naredba2;
```

if – else if … – else

```
if (uslov1)
   naredba1;
else if (uslov2)
   naredba2;
else if (…)
   …
else if (uslovN)
   naredbaN;
else
   naredba0;
```

Generalisana selekcija
if – else if … – else

izbor jedne od višе naredbi

- **prvi uslov ≠0 određuje naredbu koja se izvršava**
- **ako su svi uslovi =0 izvršava se naredba u else-u**

if – else if ... – else

poredak pisanja je važan!

- **izvršava se samo jedna naredba ako je zadovoljno više uslova**
- **kao i zbog eventualnih bočnih efekata**

ugneždavanje osnovnih selekcija

if – else if ... – else

```
if (bodovi>90)
   ocena=10;
else if (bodovi>80)
   ocena=9;
…
else if (bodovi>50)
   ocena=6;
else
   ocena=5;
```

Selekcija sa skretnicom switch

- **niz naredbi u {}**
  (sastoji se iz podnizova naredbi za pojedine opcije bez {})

- **selektorski celobrojni izraz u ()**
  određuje ulaz
  odakle počinje izvršavanje

switch

```
switch (izraz)
{
   case v1: niz_naredbi_1;
   case v2: niz_naredbi_2;
   …
   default: niz_naredbi_0;
   …
   case vN: niz_naredbi_N;
}
```

switch

- **može isti ulaz za više vrednosti**
- **default (kad vrednost izraza nije ni jedna od zadatih)**
- **default samo jedan, a može i da izostane**

switch

- **poredak je vrlo važan (za razliku od Pascal-a)**
- **ako se na kraju svakog podniza ubaci break – ekvivalentno sa Pascal case**

switch

```
switch(ocena) {
   case 10 : printf("Odlicno!\n" ); break;
   case 9 : case 8 : printf("Uspesno uradjeno\n" );
   break;
   case 7 : printf("Prolaz!\n" ); break;
   case 6 : printf("Moglo je i bolje\n" ); break;
   default : printf("Nepostojeca ocena\n" );
 }
```

Ciklusi

- **kontrolisano ponavljanje proste ili složene naredbe u zavisnosti od uslova**

- **postoji i način eksplicitnog završavanja ciklusa**

Ciklusi

kontrolisano ponavljanje proste ili složene naredbe zavisno od uslova

- **<u>while</u> - osnovni ciklus sa izlaskom na vrhu**
- **<u>for</u> - generalisani ciklus sa izlaskom na vrhu**
- **<u>do</u> - ciklus sa izlaskom na dnu**

while

osnovni ciklus sa izlaskom na vrhu

while (uslov) naredba

uslov ostanka (pre naredbe):

- **ako je ≠0 izvršava se naredba**
- **ako je =0 napušta se petlja**
- **uslov je tipa _Bool**

0, 1 ili više iteracija

while

- **Primer:**

```
S=0; i=0;
while (i<n) {
    s+=a[i]*b[i];
    i++;
}
```

do – while

ciklus sa izlaskom na dnu

do naredba while (uslov) ;

uslov ostanka (nakon naredbe):

- **ako je ≠0, opet se izvršava naredba**
- **ako je =0, napušta se petlja**
- **uslov tipa _Bool**

1 ili više iteracija

do – while

- **Primer:**

```
S=0; i=0;
do {
   s+=a[i]*b[i];
   i++;
} while (i<n);
```

- **radi pogrešno za n <= 0**

for

generalisani ciklus sa izlaskom na vrhu

for (izraz1; uslov2; izraz3) naredba

- **izraz1**
  početne radnje (samo jednom)
- **uslov2**
  ≠0 uslov nastavljanja ciklusa (_Bool)
- **izraz3**
  završne radnje (posle iteracije)

for

```
for (izraz1; izraz2; izraz3) {…}
```

⇔

```
izraz1;
while (uslov2) {
    …
    izraz3;
}
```

for

- **centralizovan kontrolni mehanizam ⇒ dobra preglednost (pogotovo važno u ugneždenim petljama)**
- **sva tri izraza opciono (ali ; moraju ostati)**
- **ako nema uslov2 ⇒ uslov ispunjen**

for

- **for (; ;) ;**
  beskonačna petlja

- **for (; ;) {...}**
  može se izaći forsirano iz bloka

- **moguća greška:**
  ; iza zagrade ⇒ prazno telo

for

- **pogodan za <u>brojačke cikluse</u>, mada je opštiji**

```
for (i=pocetak; i<=kraj; i += korak) {}
```

(korak>0)

```
for (i=pocetak; i>=kraj; i += korak) {}
```

(korak<0)

for

- **univerzalno rešenje**

```
for (i=poc; k>0?i<=kraj:i>=kraj; i+=k){
…
}
```

- **kao brojač može se koristiti i znakovna promenljiva**

for

- **skalarni proizvod:**

```
for (s=0, i=0; i<n; i++) s+=a[i]*b[i];

for (s=0, i=0; i<n; s+=a[i]*b[i], i++);
```

- **ne treba da izrazi1 i 3 sadrže radnje koje se ne odnose na održavanje ciklusa**

for

razlike u odnosu na Pascal for:

- **opštija (promenljivi korak, može i proizvoljan uslov)**
- **parametri petlje mogu biti promenjeni u samoj petlji**
- **kontrolna promenljiva zadržava vrednost po izlasku**

Skokovi

upravljačke naredbe:

- ✓ **break**
- ✓ **continue**
- ✓ **goto**

- **prenose tok izvršavanja na implicitno ili eksplicitno odredište**
- **narušavaju struktuiranost, koristiti samo kad treba**

break

izlaz iz upravljačke strukture

- **izlazak iz ciklusa (while, for, do) na prvu narebu iza ciklusa**
- **preskok preostalih naredbi u selekciji switch**

break

```
for (i=0; i<n; i++) {
    …
    for(;;) {
        …
        if (…) break;
        …
    }
}
```

- **završava se samo unutrašnji ciklus i to uslovno**

break

- **izlazak iz <u>beskonačne petlje</u>**

```
for (x=0; ; ++x) {
    …
    if (x>=3000) break;
    …
}
```

break

```
while (a>b) {
    y=f(a, b, x);
    if (y>=5*x) break;
    x=f(b,a,y);
    if (x>=5*y) break;
    a=x/2;
    b=y/2;
}
```

continue

skok na kraj ciklusa

- **preskok preostalih naredbi tekuće iteracije ciklusa**
- **zatim se izvršava opsluživanje ciklusa**

continue

- **za ugnježdene cikluse preskače se samo u najdubljem ciklusu**

- **ako je continue u switch unutar ciklusa ⇒ isto skok na kraj ciklusa**

continue

```
a=0;
while (a<n) {
   b=a/2;
   switch(z[a]){
      case 'e': z[b]<<=1; break;
      default: z[b]>>=1; continue;
      case `x':
      case 'y': z[b]=0;
   }
   ++a;
}
```

goto

skok sa proizvoljnim odredištem

goto labela ;

- **skok na labelu (identifikator)**
- **moguć skok na bilo koju labelu u funkciji**
- **ne treba uskakati u kontrolnu strukturu, može iz nje**

goto

- **može se iskakati iz proizvoljne dubine ugneždenih struktura**
- **nije potrebna, sve može bez nje**
- **ne preporučuje se**

goto

```
for (…) {
    …
    for () {
        …
        if (…) goto greska;
        …
    }
    …
}
…
greska:  /* obrada greske */
```