Uvod u računarstvo
8. predavanje

Saša Singer

singer@math.hr
web.math.hr/~singer

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Operatori i izrazi (drugi dio):
  - Relacijski i logički operatori.
  - Operatori nad bitovima.
  - Uvjetni operatori `?` i `:`.
  - Operator zarez `,`.
  - Prioriteti i redoslijed izračunavanja.

Relacijski operatori

Jezik C ima šest relacijskih operatora:

- `<` , `<=` , `>` , `>=` , `==` , `!=` ,

podijeljenih u dvije grupe po prioritetu.

Standardni ili "uređajni" relacijski operatori su:

Operator	Značenje
`<`	strogo manje
`<=`	manje ili jednako
`>`	strogo veće
`>=`	veće ili jednako

Oni imaju isti prioritet, niži od prioriteta aritmetičkih i unarnih operatora.

Operatori jednakosti

Operatori jednakosti su:

Operator	Značenje
==	jednako
!=	različito

Oni imaju zasebnu prioritetnu grupu s

- manjim prioritetom od standardnih relacijskih operatora.

Asocijativnost relacijskih operatora i operatora jednakosti je

- slijeva udesno, L $\rightarrow$ D.

Relacijski operatori – primjer

Primjer.

```
int a = 1, b = 20, limit = 100;
int rezultat;

rezultat = a < b;
  /* rezultat = 1 (istina) jer 1 < 20 */
rezultat = a == b;
  /* rezultat = 0 (laz) jer 1 != 20 */
rezultat = (a + 10) >= limit;
  /* rezultat = 0 (laz) jer (1 + 10) < 100 */
```

Logički izrazi

Kao što smo vidjeli u prethodnom primjeru, pomoću šest relacijskih operatora formiraju se tzv. logički izrazi. Njihova vrijednost je

- istina (`1`) ili laž (`0`).

C90 nema poseban logički tip, pa je vrijednost logičkih izraza tipa `int`.

Primjer: za `i = 1, j = 2, k = 4` imamo

Izraz	Istinitost	Vrijednost
`i < j`	istinito	1
`(i + j) >= k`	neistinito	0
`i == 2`	neistinito	0
`k != i`	istinito	1

Logički izrazi (nastavak)

Prioritet relacijskih operatora niži je od prioriteta aritmetičkih operatora.

Zato je izraz

```
i >= 'A' - 'a' + 1
```

ekvivalentan s

```
i >= ('A' - 'a' + 1)
```

Prioritet je "podešen" tako da zagrade ne treba pisati (iako je katkad korisno, za čitljivost).

Logički operatori

Složeniji logički izrazi tvore se pomoću logičkih operatora:

Operator	Značenje
&&	logičko I
||	logičko ILI
!	logička negacija (unarno)

Operandi logičkih operatora su logičke vrijednosti (najčešće logički izrazi), s tim da se

- svaka cjelobrojna vrijednost različita od nule interpretira kao istina,
- a nula se interpretira kao laž.

Vrijednost složenog logičkog izraza je 0 (laž) ili 1 (istina).

Logički operatori (nastavak)

Svaki od logičkih operatora ima svoj poseban prioritet.

- Unarni operator `!` (logička negacija) spada u istu grupu kao i ostali unarni operatori.
  - Asocijativnost je, također, D → L.
- Binarni operatori `&&` i `||` imaju niži prioritet od relacijskih i aritmetičkih operatora.
  - Asocijativnost je uobičajena, slijeva udesno, L → D.
- Operator `&&` (logičko i, što odgovara množenju) ima viši prioritet od `||` (logičko ili, što odgovara zbrajanju).

Logički operatori (nastavak)

Primjer. Ako je

- `i > 1` i `c =='t'` istinito, a
- `j < 6` lažno,

onda je:

Izraz	Istinitost	Vrijednost
`i > 1 || j < 6`	istinito	1
`i > 1 && j < 6`	neistinito	0
`!(i > 1)`	neistinito	0
`i > 1 || (j < 6 && c != 't')`	istinito	1

Uočite da pišu samo zagrade koje zaista hoćemo (za promjenu prioriteta).

Logički operatori – primjer 1

Primjer.

```
int a = 0, b = 10, c = 100, d = 200;
int rezultat;

rezultat = !(c < d);
  /* rezultat = !(100 < 200) = 0 */
rezultat = (a - b) && 1;
  /* rezultat = 1 jer -10 && 1 = 1 */
rezultat = d || b && a;
  /* rezultat = 1 jer 200 || (10 && 0) */
```

Logički operatori – primjer 2

Primjer. Operator negacije ! može koristiti i ovako:

```
if (!zadovoljava) ...
```

što je ekvivalentno s

```
if (zadovoljava == 0) ...
```

Prvi oblik je zgodan za logičke testove ("ako ne zadovoljava, onda ... ").

Drugi oblik je čitljiviji za numeričke testove.

Tablica prioriteta (nepotpuna)

Kategorija	Operatori	Asoc.
unarni	`! ++ -- - * & (type) sizeof`	D → L
aritm. mult.	`* / %`	L → D
aritm. adit.	`+ -`	L → D
relacijski	`< <= > >=`	L → D
rel. jednakost	`== !=`	L → D
logičko I	`&&`	L → D
logičko ILI	`||`	L → D
pridruživanje	`=`	D → L

Skraćeno izračunavanje logičkih izraza

Logički izrazi koji se sastoje od pojedinačnih logičkih izraza

- povezanih binarnim operatorima `&&` i `||`

izračunavaju se slijeva nadesno (L $\rightarrow$ D).

Važno: u C-u se koristi tzv. skraćeno izračunavanje takvih izraza. To znači da se

- izraz prestaje računati onog trena kad njegova vrijednost postane poznata.

Standardni primjeri. U izrazima:

- `op1 || op2` — ako je `op1` istinit, `op2` se ne računa.
- `op1 && op2` — ako je `op1` lažan, `op2` se ne računa.

Izračunavanje logičkih izraza (nastavak)

Skraćeno izračunavanje logičkih izraza se vrlo često koristi u programima. Pri tome treba biti oprezan

- kojim redom pišemo pojedine izraze,

jer može doći do grešaka koje se relativno teško otkrivaju!

Primjer. U odsječku kôda

```
char x[128];
for (i = 0; i < 128 && x[i] != 'a'; ++i) {
    ...
}
```

za `i = 128` (na kraju petlje), neće doći do ispitivanja `x[i] != 'a'`, što je korektno, jer `x[128]` ne postoji!

Izračunavanje logičkih izraza (nastavak)

Za razliku od prethodnog, kôd s obratnim poretkom izraza

```
char x[128];
for (i = 0; x[i] != 'a' && i < 128; ++i) {
                /* GRESKA */
    ...
}
```

za `i = 128`,

- prvo ispituje je li `x[128]` različito od `'a'`,

što je greška ("gazimo" po memoriji).

Operatori – zadaća

Primjer.

```
#include <stdio.h>
#define PR(x) printf("%d\n",(x));

int main(void){
    int x, y, z;
    x = -4 % 4 / 4 + -4;          PR(x);
    y = 4 / -x ++ -4;             PR(x); PR(y);
    y *= z = x + 4 == 4 / -y;     PR(y); PR(z);
    x = x || y && --z;            PR(x); PR(y); PR(z);
    PR(++x && ++y || ++z) ;       PR(x); PR(y); PR(z);
    return(0); }
```

Rješenje: -4, -3, -3, -3, 1, 1, -3, 1, 1, 2, -2, 1.

Operatori nad bitovima

Operatori nad bitovima mogu se primijeniti na cjelobrojne tipove podataka `char`, `short`, `int` i `long` i djeluju na bitove unutar varijable:

Operator	Značenje
`&`	logičko I bit po bit
`|`	logičko ILI bit po bit
`^`	ekskluzivno logičko ILI bit po bit
`<<`	lijevi pomak
`>>`	desni pomak
`~`	1-komplement (negacija bit po bit)

Svi operatori osim zadnjeg su binarni, a `~` je unarni.

Logički operatori nad bitovima

Operatori `&`, `^` i `|` uzimaju dva operanda i vrše operacije na bitovima koji se nalaze na odgovarajućim mjestima.

Definicije operacija dane su u sljedećoj tablici:

`b1`	`b2`	`b1 & b2`	`b1 ^ b2`	`b1 | b2`
1	1	1	0	1
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Prioritet je redom: `&`, `^` pa `|`, ispod relacijskih jednakosti, iznad logičkog `&&`. Asocijativnost je L $\rightarrow$ D.

Oprez! C nema binarnih konstanti!

Logički operatori nad bitovima (nastavak)

Primjer:

Logičko I:

```
      a  =  0x0003   /* = 0000 0000 0000 0011 */
      b  =  0x0009   /* = 0000 0000 0000 1001 */
  a & b  =  0x0001   /* = 0000 0000 0000 0001 */
```

Logičko ILI:

```
      a  =  0x0003   /* = 0000 0000 0000 0011 */
      b  =  0x0009   /* = 0000 0000 0000 1001 */
  a | b  =  0x000b   /* = 0000 0000 0000 1011 */
```

Logički operatori nad bitovima i 1-komplement

Ekskluzivno logičko ILI:

```
      a   =   0x0003   /* = 0000 0000 0000 0011 */
      b   =   0x0009   /* = 0000 0000 0000 1001 */
  a ^ b   =   0x000a   /* = 0000 0000 0000 1010 */
```

Unarni operator 1-komplement (`~`) djeluje tako da jedinice u zapisu pretvara u nule i obratno, nule u jedinice.

Primjer:

1-komplement:

```
   a   =   0x0c03   /* = 0000 1100 0000 0011 */
  ~a   =   0xf3fc   /* = 1111 0011 1111 1100 */
```

Operatori pomaka

Operatori pomaka `<<` i `>>` pomiču binarni zapis broja nalijevo ili nadesno.

- Operatori pomaka nalijevo `<<` i nadesno `>>` uzimaju dva operanda:
  - prvi operand mora biti cjelobrojni tip nad kojim se operacija vrši,
  - a drugi operand je broj bitova za koji treba izvršiti pomak (tipa `unsigned int`).
- Drugi operand ne smije premašiti broj bitova u prvom operandu.

Prioritet operatora `<<` i `>>` je isti, ispod aritmetičkih aditivnih, iznad relacijskih. Asocijativnost je L $\rightarrow$ D.

Operatori pomaka (nastavak)

- `<<` pomiče bitove ulijevo i to ne ciklički, tj. najznačajniji se bitovi gube, a zdesna se dodaju nule.
- `>>` pomiče bitove udesno i to ne ciklički, tj. najmanje značajni bitovi se gube.
  - Ako se pomak vrši na varijabli tipa `unsigned`, slijeva se dodaju nule.
  - Ako je varijabla cjelobrojnog tipa s predznakom, rezultat ovisi o implementaciji. Većina prevoditelja na lijevoj strani uvodi bit predznaka, dok drugi popunjavaju prazna mjesta nulama.

Operatori pomaka (nastavak)

Primjer. `b = a << 6` radi sljedeće:

```
      a   =   0x60ac   /* = 0110 0000 1010 1100 */
a << 6    =   0x2b00   /* = 0010 1011 0000 0000 */
```

Sve se bitovi pomiču 6 mjesta ulijevo.

Primjer. `b = a >> 6` radi sljedeće:

```
      a   =   0x60ac   /* = 0110 0000 1010 1100 */
a >> 6    =   0x0182   /* = 0000 0001 1000 0010 */
```

Ako je `a` cjelobrojnog tipa s predznakom rezultat ovisi o implementaciji — na lijevoj strani uvodi se bit predznaka ili nule.

Jednostavni primjer

Primjer. Ako je u programu definirano:

```
int x = 3;    /* ... 0011 */
int y = 5;    /* ... 0101 */
```

onda bitovni operatori daju sljedeći rezultat:

```
~x = -4          /* x + ~x + 1 = 0, v. UuR */
x & y = 1        /* ... 0001 */
x | y = 7        /* ... 0111 */
x ^ y = 6        /* ... 0110 */
x << 2 = 12      /* ... 1100 */
y >> 2 = 1       /* ... 0001 */
```

Maskiranje

Logički operatori najčešće služe maskiranju pojedinih bitova u operandu.

- Logičko I kao služi postavljanju određenih bitova na 0.
- Logičko ILI služi postavljanju određenih bitova na 1.
- Ekskluzivno ILI možemo koristiti za postavljanje određenih bitova na `1` ako su bili `0` i obratno.

Primjer. Postavimo deseti najmanje značajan bit na nulu.

Ako u varijabli `a` želimo postaviti neki bit na nulu, dovoljno je napraviti logičko I s varijablom `mask` koja na tom mjestu ima nulu, a na svim ostalim jedinice.

Maskiranje (nastavak)

Varijablu `mask` je najlakše napraviti tako da se:

- prvo postavi na 1 (najmanje značajni bit je 1),
- izvrši pomak 9 mjesta ulijevo, čime je dobivena jedinica na 10. mjestu, a sve ostalo su nule,
- varijabla `mask` se 1-komplementira.

```
int a = 25;
unsigned mask;

mask = ~(1 << 9);
a = a & mask;  /* a &= mask */
```

Maskiranje (nastavak)

Primjer. Šest najmanje značajnih bitova treba iz varijable `a` kopirati u `b`. Sve ostale bitove u `b` treba staviti na `1`.

Prvo definiramo varijablu `mask` koja ima šest najmanje značajnih bitova jednakih `0`, a ostale `1`. Logički ILI s `mask` izdvaja najmanje značajne bitove u `a` i postavlja vodeće bitove u `b` na `1`.

```
mask = 0xffc0    /* = 1111 1111 1100 0000 */
b = a | mask;
```

Ova operacija ovisi o duljini tipa za `int`, no to se može izbjeći korištenjem 1-komplementa.

```
mask = ~0x3f    /* = ~11 1111 */
```

Složeniji primjer

Primjer. Program koji ispisuje binarni zapis cijelog broja tipa `int`.

```
#include <stdio.h>
int main(void) {
    int a, b, i, nbits;
    unsigned mask;

    nbits = 8 * sizeof(int);
        /* duljina tipa int */
    mask = 0x1 << (nbits - 1);
        /* 1 na najznacajnijem mjestu */
    printf("\nUnesite cijeli broj: ");
    scanf("%d", &a);
```

Složeniji primjer (nastavak)

```
    for (i = 1; i <= nbits; ++i) {
        b = (a & mask) ? 1 : 0;
        printf("%d", b);
        if (i % 4 == 0) printf(" ");
        mask >>= 1;
    }
    printf("\n");
    return 0;
}
```

Tablica prioriteta operatora (nepotpuna)

Kategorija	Operatori	Asoc.
unarni	`! ~ ++ -- - * &`	D → L
aritm. mult.	`* / %`	L → D
aritm. adit.	`+ -`	L → D
op. pomaka	`<< >>`	L → D
relacijski	`< <= > >=`	L → D
rel. jednakost	`== !=`	L → D
bitovni I	`&`	L → D
bitovni eks. ILI	`^`	L → D
bitovni ILI	`|`	L → D

Tablica prioriteta operatora (nepotpuna)

Kategorija	Operatori	Asoc.
logičko I	`&&`	L → D
logičko ILI	`||`	L → D
pridruživanje	`= += -= *= /= %=`	D → L
pridruživanje (nast.)	`&= ^= |= <<= >>=`	D → L

Složeni operatori pridruživanja

Složeni operatori pridruživanja su:

- `+=`, `-=`, `*=`, `/=`, `%=` (aritmetički).
- `<<=`, `>>=`, `&=`, `^=`, `|=` (bitovni).

Općenito, izraz oblika

```
izraz1 op= izraz2;
```

gdje je `op` jedna od operacija `+`, `-`, `*`, `/`, `%`, `<<`, `>>`, `&`, `^`, `|`, ekvivalentan je s

```
izraz1 = izraz1 op izraz2;
```

Tj. lijeva strana je ujedno i prvi operand.

Složeni operatori pridruživanja (nastavak)

Ovi složeni operatori spadaju u istu prioritetnu grupu s operatorom pridruživanja = i imaju istu asocijativnost zdesna nalijevo, D → L.

Primjer:

Izraz	Ekvivalentan izraz
`i += 5`	`i = i + 5`
`i -= j`	`i = i - j`
`i *= j + 1`	`i = i * (j + 1)`
`i /= 4`	`i = i / 4`
`i %= 2`	`i = i % 2`
`i <<= 2`	`i = i << 2`
`i &= k`	`i = i & k`

Uvjetni operator ? :

Uvjetni izraz je izraz oblika

```
izraz1 ? izraz2 : izraz3;
```

U njemu se prvo izračunava `izraz1`.

- Ako je on istinit (različit od nule), izračunava se `izraz2` i on postaje vrijednost čitavog uvjetnog izraza.
- Ako je `izraz1` lažan (jednak nuli), izračunava se `izraz3` i on postaje vrijednost čitavog uvjetnog izraza.

"Paket" od dva simbola `? :` je zapravo ternarni operator, tj. ima 3 operanda.

Uvjetni operator ? : (nastavak)

Primjer:

```
double a,b;
...
(a < b) ? a : b;
```

je izraz koji daje manji od brojeva `a` i `b`.

Vrijednost uvjetnog izraza može se pridružiti nekoj varijabli:

```
double a, b, min;
...
min = (a < b) ? a : b;
```

Korisno za razne inicijalizacije!

Operator zarez ,

Operator zarez , separira dva izraza. Izrazi separirani zarezom izračunavaju se slijeva nadesno i rezultat čitavog izraza je vrijednost desnog izraza.

Primjer:

```
i = (i = 3, i + 4);
```

daje rezultat `i = 7`. Operator zarez uglavnom se koristi u `for` naredbi.

Prioritet operatora , je niži od operatora pridruživanja (tj. na dnu tablice prioriteta). Zato su nužne zagrade na desnoj strani operatora = u gornjem primjeru.

Asocijativnost je, naravno (po ideji), slijeva nadesno, L $\rightarrow$ D.

Tablica prioriteta operatora (nepotpuna)

Uvjetni operator ima nizak prioritet, tako da zagrade oko prvog, drugog i trećeg izraza najčešće nisu potrebne.

Kategorija	Operatori	Asoc.
unarni	`! ~ ++ -- - * &`	D → L
aritm. mult.	`* / %`	L → D
aritm. adit.	`+ -`	L → D
op. pomaka	`<< >>`	L → D
relacijski	`< <= > >=`	L → D
rel. jednakost	`== !=`	L → D
bitovni I	`&`	L → D
bitovni eks. ILI	`^`	L → D
bitovni ILI	`|`	L → D

Tablica prioriteta operatora (nepotpuna)

Kategorija	Operatori	Asoc.
logičko I	`&&`	L → D
logičko ILI	`||`	L → D
uvjetni	`?  :`	D → L
pridruživanje	`= += -= *= /= %=`	D → L
pridruživanje (nast.)	`&= ^= |= <<= >>=`	D → L
op. zarez	`,`	L → D