Programiranje 1
7. predavanje

Saša Singer

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

PMF – Matematički odjel, Zagreb

Sadržaj predavanja

- Operatori i izrazi (drugi dio):
  - `sizeof` operator.
  - Relacijski operatori.
  - Logički operatori.
  - Skraćeno računanje logičkih izraza.
  - Operatori nad bitovima.
  - Operator pridruživanja i složeni operatori pridruživanja.
  - Uvjetni operatori `?` i `:` .
  - Operator zarez `,` .
  - Tablica prioriteta i asocijativnosti operatora.

Informacije

Predavanje od petka, 9. 10., održavamo sutra

- u subotu, 24. 10., od 10–12 u (003).

Termin prvog kolokvija je (službeno):

- utorak, 10. 11., u 12 sati.

Termin za konzultacije je:

- ponedjeljak, 14–16 sati.

Informacije — Računi i zadaće

Ne zaboravite da treba:

- otvoriti korisnički račun u Računskom centru.
- Računi se "preuzimaju" u centru, utorkom i četvrtkom, od 12:30 do 15 sati.

Promijenite password!

Nadalje, treba:

- obaviti prijavu i dobiti potvrdu prijave

u aplikaciji za tzv. "domaće zadaće", na web–adresi

`http://degiorgi.math.hr/prog1/ku/`

Informacije — nastavak

Bitno: Prilikom prijave za "ku",

- svoje podatke trebate upisati korektno,

što (između ostalog) znači i

- korištenje hrvatskih slova u imenu i prezimenu!

Studenti koji su upisali "czsdj" varijantu imena i prezimena neka se jave e-mailom asistentu V. Šegi na adresu

`vsego@math.hr`

i napišu

- svoj JMBAG i ispravno ime i prezime.

Informacije — Upozorenje

Upozorenje — vezano uz prijave za zadaće:

- trenutni broj uspješno prijavljenih studenata još uvijek je zabrinjavajući — oko 210.

Pogledajte popis studenata na webu!

Lijepo molim, "ne šalite" se, uspješna prijava je

- nužan preduvjet za izlazak na kolokvij.

To pravilo se ne mijenja! Rok za prijavu je 48 sati prije kolokvija.

Operatori i izrazi (prvi dio) ponavljanje

Sadržaj

- Operatori i izrazi (prvi dio) — ponavljanje:
  - Aritmetički operatori.
  - Pretvaranje tipova u aritmetičkim izrazima.
  - Unarni operatori inkrementiranja i dekrementiranja.

Aritmetički operatori

U programskom jeziku C postoji 5 aritmetičkih operatora:

operator	značenje
`+`	zbrajanje, unarni plus
`-`	oduzimanje, unarni minus
`*`	množenje
`/`	dijeljenje
`%`	ostatak (modulo)

Operatori `-` i `+` imaju dva različita značenja,

- ovisno o tome kako se piše — obzirom na operand.

Aritmetički operatori (nastavak)

Operator promjene predznaka `-` je unarni operator i

- piše se ispred operanda

(tzv. prefiks notacija).

```
-operand
```

Slično vrijedi i za unarni `+` (ne mijenja predznak).

Ostali aritmetički operatori su binarni i

- pišu se između dva operanda

(tzv. infiks notacija).

```
operand_1 operacija operand_2
```

Aritmetički operatori (nastavak)

Djeluju na numeričke operande raznih tipova. Operandi mogu biti:

- nekog cjelobrojnog tipa,
- nekog realnog tipa,
- znakovnog tipova (`char` se prikazuje kao cijeli broj).

Problem: Kad operandi nisu istog tipa, kojeg tipa je rezultat?

- Tada dolazi do konverzije (pretvaranja) tipova po određenim pravilima (v. malo kasnije).

Za početak, pogledajmo kako radi cjelobrojno dijeljenje!

Cjelobrojno dijeljenje

Operacija dijeljenja `/`, u slučaju kad su

- oba operanda cjelobrojna,

daje cjelobrojan rezultat (operacija div od ranije).

Po C99 standardu, rezultat se uvijek dobiva zaokruživanjem kvocijenta prema nuli. Dakle, vrijedi:

```
3/2 = 1,    -3/2 = -1.
```

Po C90 standardu, to vrijedi za pozitivne operande. Inače, ako je bar jedan operand negativan, rezultat ovisi o implementaciji.

Napomena: Ako je bar jedan operand realan broj, dijeljenje je uobičajeno dijeljenje realnih brojeva, pa je `3.0/2 = 1.5`.

Cjelobrojni ostatak (modulo)

Operator `%` (modulo) djeluje samo na cjelobrojnim operandima i kao rezultat daje

- cjelobrojni ostatak pri cjelobrojnom dijeljenju operanada.

(operacija mod od ranije).

Primjer: za `x = 10` i `y = 3` dobivamo

```
x / y = 3,     x % y = 1.
```

Ostatak se računa tako da uvijek vrijedi (osim za `y == 0`):

```
(x / y) * y + x % y == x
```

Dakle, ostatak ima predznak prvog operanda.

Konverzije u aritmetičkim izrazima

Kad aritmetički operator ima dva operanda različitog tipa, onda dolazi do

- konverzije (ili pretvaranja) tipova.

U pravilu se:

- prije operacije, operand "nižeg" ili "užeg" tipa promovira, odnosno, pretvara u "viši" ili "širi" tip,
- zatim se izvršava operacija — sad na operandima istog tipa,
- rezultat operacije ima taj isti (zajednički) tip.

Ovo pretvaranje se radi onim redom kojim se izvršavaju operacije u izrazu, tj. po prioritetu operacija (v. kasnije).

Konverzije u aritmetičkim izrazima (nastavak)

Na primjer:

- Operandi tipa `char` i `short` (s predznakom ili bez njega) automatski se pretvaraju u tip `int` ili `unsigned int`, i to prije svake aritmetičke operacije.
- Ako su operandi tipa `float` i `double`, onda se, prije izračunavanja,
  - `float` pretvara u `double`.
- Ako je jedan od operanada realnog tipa (`float`, `double`), a drugi cjelobrojnog (`int`, `unsigned`, `long`, ...), onda se
  - cjelobrojni tip pretvara u realni.

Konverzije kod pridruživanja

U operaciji pridruživanja dolazi do konverzije, ako tip lijeve strane nije isti kao tip desne strane. Tada se:

- operand na desnoj strani konvertira u tip operanda na lijevoj strani.

Pri tome može doći do gubitka informacije,

- ako se širi tip konvertira u uži.

Najčešći primjer je pretvaranje realnog u cjelobrojni tip.

- To se radi "odbacivanjem", tj. zaokruživanjem prema nuli.

Na primjer, ako je `x` varijabla tipa `float` i `n` varijabla tipa `int`, prilikom pridruživanja `n = x` doći će do odsjecanja decimala u broju `x`.

Operatori inkrementiranja i dekrementiranja

Operatori inkrementiranja i dekrementiranja pišu se u prefiks i postfix obliku.

Značenje: `x++` i `++x` znači "povećaj `x` za 1", a `x--` i `--x` znači "smanji `x` za 1".

Primjer. Razlika između prefiksa i postfiksa — kad se izvrši operacija inkrementiranja/dekrementiranja!

```
/* prefiks */                    /* postfiks */
   int x, z;                        int x, z;
   x = 3;                           x = 3;
   z = ++x - 2;                     z = x++ - 2;
                  /* rezultati */
/* x = 4, z = 2 */               /* x = 4, z = 1 */
```

Operatori i izrazi (nastavak)

Sadržaj

- Operatori i izrazi (drugi dio):
  - `sizeof` operator.
  - Relacijski operatori.
  - Logički operatori.
  - Skraćeno računanje logičkih izraza.
  - Operatori nad bitovima.
  - Operator pridruživanja i složeni operatori pridruživanja.
  - Uvjetni operatori `?` i `:` .
  - Operator zarez `,` .
  - Tablica prioriteta i asocijativnosti operatora.

Operator `sizeof`

Operator `sizeof` vraća veličinu svog operanda u bajtovima.

- Operand može biti izraz ili tip podatka.

Primjer:

```
int i;
double x;
printf("Velicina tipa int    = %d\n",
       sizeof(i));
printf("Velicina tipa double = %d\n",
       sizeof(x));
```

Na standardnim računalima rezultati su `4` i `8`.

Ovdje je operand izraz (sastavljen od jedne varijable).

Operator `sizeof` (nastavak)

Isti efekt postižemo ako `sizeof` primijenimo na tip podatka:

```
printf("Velicina tipa int    = %d\n",
       sizeof(int));
printf("Velicina tipa double = %d\n",
       sizeof(double));
```

Napomena: `sizeof(char)` je uvijek jednak 1.

Kod složenijih tipova podataka dobivamo ukupan broj bajtova koji podatak (tog tipa) zauzima.

Na primjer, za polje dobivamo "veličinu" cijelog polja (ako smo u funkciji u kojoj je rezervirana memorija za polje, inače ne — v. kasnije).

Operator `sizeof` (nastavak)

Primjer:

```
char tekst[] = "Program";
printf("Broj znakova u varijabli tekst = %d\n",
        sizeof(tekst));
```

ispisuje

```
Broj znakova u varijabli tekst = 8
```

Operator `sizeof` vraća cjelobrojnu vrijednost (bez predznaka) koja ovisi o implementaciji.

Pripadni tip definiran je u datoteci zaglavlja `<stddef.h>` i zove se `size_t` (tip za "veličine", odnosno, "duljine" objekata).

Tablica prioriteta operatora (nepotpuna)

Operator `sizeof` je

- unarni operator, asocijativnost D → L,

i pripada istoj prioritetnoj grupi kao i ostali unarni operatori.

Kategorija	Operatori	Asoc.
unarni	`++ -- + - * & (type) sizeof`	D → L
aritm. mult.	`* / %`	L → D
aritm. adit.	`+ -`	L → D
pridruživanje	`=`	D → L

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 */
```

Tablica prioriteta operatora (nepotpuna)

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

Logički izrazi

Relacijskim operatorima 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 (unarna)

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

- svaka cjelobrojna vrijednost različita od nule interpretira kao istina,
- a samo 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')`	neistinito	0

Uočite da u tablici pišu samo one 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 = 0, jer !(100 < 200) = !1 =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 se 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 operatora (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 → 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. Koristi složene operatore pridruživanja (v. malo iza).

```
#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 (ne 0), 1, 2, -2, 1.

Operatori nad bitovima

Operatori nad bitovima mogu se primijeniti na cjelobrojne tipove podataka `char`, `short`, `int` i `long`, a djeluju na bitove u prikazu podatka:

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. Uzmimo da su `a` i `b` cjelobrojne varijable duljine 16 bitova (za lakše "crtanje").

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 (nastavak)

Ekskluzivno logičko ILI:

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

Logički operator 1–komplement

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

Primjer. Neka je `a` cjelobrojna varijabla duljine 16 bitova (za lakše "crtanje").

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 cjelobrojnog tipa i nad tim operandom se vrši operacija,
  - 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 podatku tipa `unsigned`, slijeva se dodaju nule.
  - Ako je podatak 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 */
```

Svi bitovi pomiču se 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. ranije. */
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 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, v. malo kasnije */
```

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 */
```

Tablica prioriteta operatora (nepotpuna)

Kategorija	Operatori	Asoc.
unarni	`!` `~` `++` `--` `+` `-` `*` `&` `(type)` `sizeof`	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

Operatori pridruživanja

Osnovni operator pridruživanja je `=`. Prioritet mu je

- niži od većine ostalih operatora (`,` je izuzetak).

To je zato da naredba pridruživanja, oblika

```
varijabla = izraz;
```

prvo izračuna izraz na desnoj strani, a onda tu vrijednost pridruži varijabli.

Primjer:

```
i = 2;
a = 3.17;
c = 'm';
```

Operatori pridruživanja (nastavak)

Pridruživanje `varijabla = izraz` je ujedno i izraz.

- Vrijednost tog izraza je vrijednost varijable na lijevoj strani, nakon što se izvrši pridruživanje.

Stoga pridruživanje može biti dio složenijeg izraza.

Primjer:

```
while ((a = x[i]) != 0) {
    ...
    ++i;
}
```

U testu `while` naredbe prvo se `x[i]` pridruži varijabli `a`, a onda se testira je li dobiveni izraz (a to je vrijednost varijable `a`) različit od nule.

Operatori pridruživanja (nastavak)

Ovakva mogućnost može uzrokovati teško uočljive greške.

Primjer. Ako napišemo

```
if (varijabla = izraz) ...;
```

umjesto uobičajenog

```
if (varijabla == izraz) ...;
```

onda će se u prvoj `if` naredbi prvo vrijednost izraza pridružiti varijabli, a zatim će se izvršiti tijelo `if` naredbe ako je varijabla različita od nule.
U drugoj `if` naredbi vrijednost varijable se ne mijenja, već se samo uspoređuje s vrijednošću izraza. Tijelo `if` naredbe izvršava se ako su te dvije vrijednosti jednake.

Operatori pridruživanja (nastavak)

Asocijativnost operatora pridruživanja je

- zdesna nalijevo, D → L.

Zato operatore pridruživanja možemo ulančati:

```
varijabla_1 = varijabla_2 = ...
            = varijabla_n = izraz;
```

To znači da se

- `izraz` jednom izračuna,
- a zatim se redom pridružuje varijablama:
  `varijabla_n`, `...`, `varijabla_2`, `varijabla_1`.

Operatori pridruživanja (nastavak)

Primjer:

```
x = y = cos(3.22);
```

ekvivalentno je s

```
x = (y = cos(3.22));
```

odnosno

```
y = cos(3.22);
x = y;
```

Složeni operatori pridruživanja

Složeni operatori pridruživanja su:

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

Općenito, izraz oblika

```
izraz_1 op= izraz_2
```

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

```
izraz_1 = izraz_1 op (izraz_2)
```

Tj. lijeva strana je ujedno i prvi operand. Uočite zagrade oko `izraz_2`.

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`

Tablica prioriteta operatora (nepotpuna)

Kategorija	Operatori	Asoc.
unarni	`!` `~` `++` `--` `+` `-` `*` `&` `(type)` `sizeof`	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

Uvjetni operator ? :

Uvjetni izraz je izraz oblika

```
izraz_1 ? izraz_2 : izraz_3
```

U njemu se prvo izračunava `izraz_1`.

- Ako je on istinit (različit od nule), onda se izračunava `izraz_2` i on postaje vrijednost čitavog uvjetnog izraza. U ovom slučaju, `izraz_3` se ne računa.
- Ako je `izraz_1` lažan (jednak nuli), onda se izračunava `izraz_3` i on postaje vrijednost čitavog uvjetnog izraza. U ovom slučaju, `izraz_2` se ne računa.

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

Uvjetni operator ? : (nastavak)

Primjer. Izraz koji daje manji od brojeva `a` i `b`.

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

Vrijednost uvjetnog izraza može se pridružiti nekoj varijabli:

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

Korisno za razne inicijalizacije!

Zagrade ovdje nisu potrebne, zbog prioriteta (v. malo kasnije).

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 (v. kasnije).

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	`!` `~` `++` `--` `+` `-` `*` `&` `(type)` `sizeof`	D → L
aritm. mult.	`*` `/` `%`	L → D
aritm. adit.	`+` `-`	L → D
op. pomaka	`<<` `>>`	L → D
relacijski	`<` `<=` `>` `>=`	L → D
rel. jednakost	`==` `!=`	L → D

Tablica prioriteta operatora (nepotpuna)

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

Tablica prioriteta operatora (potpuna)

Sad smo obradili skoro sve operatore u jeziku C.

Do potpune tablice operatora fale nam još samo tzv. primarni operatori. Ima ih 4:

- `( )` — za poziv funkcije,
- `[ ]` — za pristup elementima polja,
- `.` — za pristup članovima strukture,
- `->` — za pristup članovima strukture preko pokazivača.

Primarni operatori su grupa s najvišim prioritetom, a asocijativnost je uobičajena, L → D.

Tablica prioriteta operatora (potpuna)

Kategorija	Operatori	Asoc.
primarni	`()` `[]` `->` `.`	L → D
unarni	`!` `~` `++` `--` `+` `-` `*` `&` `(type)` `sizeof`	D → L
aritm. mult.	`*` `/` `%`	L → D
aritm. adit.	`+` `-`	L → D
op. pomaka	`<<` `>>`	L → D
relacijski	`<` `<=` `>` `>=`	L → D
rel. jednakost	`==` `!=`	L → D

Tablica prioriteta operatora (potpuna)

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