Programiranje 1

Milena Vujošević Janičić

www.matf.bg.ac.rs/~milena

Programski jezik C.
Predstavljanje podataka i operacije nad njima.

Pregled

Pregled

Programski jezik C

- C — jezik opšte namene, imperativan i proceduralan
- 1972 — Denis Riči (dobitnik Tjuringove nagrade 1983. godine)
- Namena: najpre kao sistemski softver u okviru Unix-a, kasnije i za aplikativni softver
- Standardi: K & R (1978), ANSI i ISO (1989-90), C99, C11

Zdravo!

```
#include <stdio.h>

int main() {
  printf("Zdravo!\n");
  return 0;
}
```

Kvadrat unetog broja

```
#include <stdio.h>

int main() {
  int a;
  printf("Unesite ceo broj: ");
  scanf("%i", &a);
  printf("Kvadrat unetog broja je: %i", a*a);
  return 0;
}
```

Da li je broj paran?

```
#include <stdio.h>

int main() {
  int a;
  printf("Unesi broj: ");
  scanf("%d", &a);
  if (a % 2 == 0)
    printf("Broj %d je paran\n", a);
  else
    printf("Broj %d je neparan\n", a);
  return 0;
}
```

Kvadrati i koreni prvih 100 brojeva

```
#include <stdio.h>
#include <math.h>
#define N 100

int main() {
  int i;
  for (i = 1; i <= N; i++)
    printf("%3d %5d %7.4f\n", i, i*i, sqrt(i));
  return 0;
}
```

Zbir brojeva

```
#include <stdio.h>
#define N 100

int main() {
  int i = 1;
  int s = 1;
  while(s <= N) {
    i++;
    s = s+i;
  }
  printf("%d\n", i);
  return 0;
}
```

Mala slova u velika

```
#include <stdio.h>
#include <ctype.h>

int main() {
  int c;
  printf("Otkucaj recenicu (zavrsi je znakom .): ");
  do {
    c = getchar();
    putchar(toupper(c));
  } while (c != '.');
  putchar('\n');
  return 0;
}
```

Pregled

Promenljive i deklaracije

- Promenljive su osnovni objekti koji se koriste u programima.
- Svaka promenljiva mora biti deklarisane pre korišćenja.
- Promenljiva je objekat kojem je pridružen neki prostor u memoriji i u svakom trenutku svog postojanja ima vrednost kojoj se može pristupiti — koja se može pročitati i koristiti, ali i koja se (ukoliko nije traženo drugacije) može menjati.
- Imena promenljivih određena su *identifikatorima*.
- Generalno, identifikator može da sadrži slova i cifre, kao i simbol _, ali ne može počinjati cifrom.
- Ključne reči jezika C (na primer, `if`, `for`, `while`) ne mogu se koristiti kao identifikatori.

Promenljive

- U identifikatorima, velika i mala slova se razlikuju. Na primer, promenljive sa imenima `a` i `A` se tretiraju kao dve različite promenljive.
- Imena promenljivih i funkcija treba da oslikavaju njihovo značenje i ulogu u programu, ali za promenljive kao što su indeksi u petljama se obično koriste kratka, jednoslovna imena (na primer `i`).
- Ako ime promenljive sadrži više reči, onda se, radi bolje čitljivosti, te reči razdvajaju simbolom _ (na primer, `broj_studenata`) ili početnim velikim slovima (na primer, `brojStudenata`) — ovo drugo je takozvana kamilja notacija (CamelCase).

Deklaracije

- Sve promenljive moraju biti deklarisane pre korišćenja.
- Deklaracija sadrži tip i listu od jedne ili više promenljivih tog tipa, razdvojenih zarezima.

```
int broj;  /* deklaracija celog broja        */
int a, b;  /* deklaracija vise celih brojeva */
```

- U opštem slučaju nije propisano koju vrednost ima promenljiva neposredno nakon što je deklarisana.

Deklaracije

- Prilikom deklaracije može se izvršiti početna inicijalizacija.
- Moguće je kombinovati deklaracije sa i bez inicijalizacije.

```
int vrednost = 5;
int a = 3, b, c = 5;
```

- Izraz kojim se promenljiva inicijalizuje zvaćemo inicijalizator.
- Kvalifikator `const` može biti dodeljen deklaraciji promenljive da bi naznačio i obezbedio da se njena vrednost neće menjati, na primer:

```
/* ovu promenljivu nije moguce menjati */
const double GRAVITY = 9.81;
```

Tipovi

Jedan tip karakteriše:

- Vrsta podataka koje opisuje,
- Način reprezentacije,
- Skup operacija koje se mogu primeniti nad podacima tog tipa,
- Broj bitova koji se koriste za reprezentaciju (odakle sledi opseg mogućih vrednosti).

Tip int

- Cele brojeve opisuje tip `int` — od engleskog *integer*, *ceo broj*.
- Podrazumeva se da su vrednosti ovog tipa označene i reprezentuju se najčešće koristeći potpuni komplement.
- Mogu se koristiti aritmetičke operacije (npr +, -, *, /, %), relacije (npr, <, >=) itd.

Tip int

- Broj bitova nije standardom propisan, ali je propisano da se koristi najmanje šesnaest bita.
- Veličina tipa `int` je obično prilagođena konkretnoj mašini, tj 32 ili 64 bita
- Veličina podataka zavisi od *sistema*, pri čemu se pod sistemom podrazumeva i hardver računara i operativni sistem na kojem će se program izvršavati.
- Podaci o opsegu ovih (i drugih tipova) za konkretan računar i C prevodilac sadržani su u standardnoj datoteci zaglavlja `<limits.h>`.

Kvalifikatori

- `short`, `long` i od C99 `long long`
- `signed` i `unsigned`
- Specifikatori za upis i ispis:

```
%i, %d --- ceo dekadni broj
%u --- neoznacen dekadni broj
%o --- neoznacen oktalni broj
%x, %X --- neoznacen heksadekadni broj
```

- Kvalifikatori za upis i ispis `%h` i `%l`

Opseg tipova

- Konačan opseg tipova treba uvek imati u vidu

```
#include <stdio.h>

int main() {
  int a = 2000000000, b = 2000000000;
  printf("Zbir brojeva %d i %d je: %d\n", a, b, a + b);
  return 0;
}
Izlaz:
  Zbir brojeva 2000000000 i 2000000000 je: -294967296
```

Opseg tipova

- Konačan opseg tipova treba uvek imati u vidu

```
#include <stdio.h>

int main() {
  unsigned int a = 2000000000, b = 2000000000;
  printf("Zbir brojeva %u i %u je: %u\n", a, b, a + b);
  return 0;
}
Izlaz:
  Zbir brojeva 2000000000 i 2000000000 je: 4000000000
```

Tip char

- Male cele brojeve opisuje tip `char` (od engleskog *character* — *karakter, simbol, znak*).
- Mogu se primenjivati aritmetičke i operacije relacije
- Tačno jedan bajt
- Može biti označen ili neoznačen, nije propisano standardom
- Kvalifikatori signed [-128, 127] i unsigned [0, 255]
- Ovaj tip obično se koristi za brojeve koji predstavljaju kôdove karaktera.
- Najčešće se koristi ASCII kodiranje karaktera (aski kodovi su sedmobitni i imaju vrednosti od 0 do 127, što staje i u signed i unsigned char)

Tip char

- Ispis i učitavanje %c
- Standardna biblioteka sadrži mnoge funkcije i makroe za rad sa karakterskim tipom
- One su deklarisane u datoteci zaglavlja `<ctype.h>`
- Na primer `isalpha`, `isdigit`, `toupper`, `tolower`

Tip float, double i long double

- Brojevi u pokretnom zarezu: float (osnovna tačnost) i double (dvostruka tačnost), i od C99 long double (proširena tačnost)
- Nije propisano standardnom koliko ovi tipovi zauzimaju bitova
- Podaci o opsegu i detaljima ovih (i drugih tipova) za konkretan računar i C prevodilac sadržani su u standardnoj datoteci zaglavlja `<float.h>`.
- Uobičajne aritmetičke operacije (sem %) i relacije
- IEEE 754 standard, vrednosti $-\infty$, ∞, *NaN*
- *NaN* — 0.0/0.0, koren iz negativnog broja i slične nedefinisane matematičke vrednosti
- $1/\infty$ je 0, $1/0$ je ∞
- Matematičke funkcie `<math.h>`
- Ispis i upis `%f`, za long double `%lf`

Logički tip podataka

- Bez logičkog tipa, celobrojne vrednosti: 0 netačno, sve ostalo tačno
- C99 tip bool i konstante true i false
- Logičke operacije i, ili, negacija

Pregled

Konstante i konstantni izrazi

- Izrazi kombinuju promenljive i konstante korišćenjem operatora, dajući nove vrednosti
- Izrazi mogu biti promenljive, konstante, pozivi funkcija ili složeni izrazi
- Konstante su fiksne vrednosti kao, na primer, `0`, `2`, `2007`, `3.5`, `1.4e2` ili `'a'`.
- Za sve konstante i za sve izraze, pravilima jezika jednoznačno su određeni njihovi tipovi.
- Od tipova zavisi vrednost složenog izraza kao i koje je operacije moguće primeniti

Celobrojne konstante

- Celobrojne konstante su tipa int, npr 231, 9876
- Velike celobrojne konstante koje ne mogu da stanu u int a mogu u long su tipa long, ili unsigned long ukoliko ne mogu da stanu u long
- Dakle, tačan tip dekadne celobrojne konstante ne može da se odredi ako se ne znaju detalji sistema.

Celobrojne konstante

- Mogu se koristiti kvalifikatori u i U za unsigned, l i L za long, ili ul za unsigned long Na primer, 12345 je tipa int a 12345L je tip long
- Celobrojne konstante mogu biti zapisane i u oktalnom i u heksadekadnom sistemu
- Zapis konstante u oktalnom sistemu počinje cifrom `0`, a zapis konstante u heksadekadnom sistemu počinje simbolima `0x` ili `0X`. Npr `037` (oktalni zapis), `0x1f` (heksadekadni zapis)

Celobrojne konstante

- Negativne konstante ne postoje, ali se efekat može postići izrazima gde se ispred konstante navodi unarni operator -
- Kada se u tekstu programa naiđe na -123 vrednost predstavljena ovim izrazom jeste minus stodvadesettri, ali izraz nije konstanta već je sačinjen od unarnog operatora primenjenog na konstantu.
- Slično, može se navesti i operator plus, ali to nema efekta (npr. +123 je isto kao i 123).

Konstante u pokretnom zarezu

- Konstantni brojevi u pokretnom zarezu sadrže tačku ili eksponent, ili i jedno i drugo
- 123.4 ili 1e-2 ili .4 ili 5. ili -123. ili -123.2e10
- Tipovi ovih konstanti su double, osim ukoliko se za float navede kvalifikator f ili F, npr 1.23f ili l/L što označava tip long double

Karakterske konstante

- Iako se tip `char` koristi i za predstavljanje malih celih brojeva, on se prevashodno koristi za predstavljanje kôdova karaktera (najčešće ASCII kôdova).
- Direktno specifikovanje karaktera korišćenjem numeričkih kôdova nije preporučljivo.
- Umesto toga, preporučuje se korišćenje karakterskih konstanti.
- Karakterske konstante u programskom jeziku C se navode između ’’ navodnika.
- Vrednost date konstante je numerička vrednost datog karaktera u korišćenoj karakterskoj tabeli (na primer, ASCII).

Karakterske konstante

- Na primer, u ASCII kodiranju, karakterska konstanta `'0'` predstavlja vrednost 48 (koja nema veze sa numeričkom vrednošću 0),
- `'A'` je karakterska konstanta čija je vrednost u ASCII kôdu 65,
- `'a'` je karakterska konstanta čija je vrednost u ASCII kôdu 97

```
char c = 'a';
char c = 97; /* ekvivalentno prethodnom (na ASCII masinama),
                ali se ne preporucuje zbog toga sto smanjuje
                citljivost i prenosivost programa */
```

Specijalni karakteri

`\a`	alert (bell) character
`\b`	backspace
`\f`	formfeed
`\n`	newline
`\r`	carriage return
`\t`	horizontal tab
`\v`	vertical tab
`\\`	backslash
`\?`	question mark
`\'`	single quote
`\"`	double quote
`\ooo` (npr. `\012`)	octal number
`\xhh` (npr. `\x12`)	hexadecimal number

Konstantni izrazi

- Konstantni izraz je izraz koji sadrži samo konstante (na primer, `4 + 3*5`).
- Tip izraza zavisi od tipova operanada

Pregled

Operatori i izrazi

- Operatorima su predstavljene osnovne operacije i relacije koje se mogu vršiti nad podacima osnovnih tipova u jeziku C.
- Operatori se dele na osnovu svoje *arnosti* tj. broja operanada na koje se primenjuju.
- *Unarni* operatori deluju samo na jedan operand i mogu biti *prefiksni* kada se navode pre operanda i *postfiksni* kada se navode nakon svog operanda.
- *Binarni* operatori imaju dva operanda i obično su infiksni tj. navode se između svojih operanda.
- U jeziku C postoji jedan *ternarni* operator koji se primenjuje na tri operanda.

Prioritet operatora

- Izrazi mogu da obuhvataju više operatora i zagrade (i)
- Prioritet operatora — konvencija koja omogućava izostavljanje zagrada
- Na primer, vrednost konstantnog izraza `3 + 4 * 5` biće 23, jer operator * ima prioritet u odnosu na operator +.
  1. Unarni operatori imaju veći prioritet u odnosu na binarne.
  2. Postfiksni unarni operatori imaju veći prioritet u odnosu na prefiksne unarne operatore.
  3. Aritmetički operatori imaju veći prioritet u odnosu na relacijske koji imaju veći prioritet u odnosu na logičke operatore.
  4. Operatori dodele imaju veoma nizak prioritet.

Asocijativnost operatora

- *Asocijativnost operatora* definiše kojim redosledom će se izračunavati dva ista operatora ili operatora istog prioriteta kada se nađu uzastopno u istom, nezagrađenom izrazu.
- Obično se razlikuju *leva asocijativnost*, kada se izraz izračunava sleva na desno i *desna asocijativnost*, kada se izraz izračunava zdesna na levo.
- Većina operatora ima levu asocijativnost (najznačajniji izuzeci su prefiksni unarni operatori i operatori dodele).

Operator dodele

```
broj_studenata = 80;
broj_grupa     =  2;
```

- U dodeljivanju vrednosti, sa leve strane operatora dodele mora biti l-vrednost (promenljiva, element niza ili memorijska lokacija)
- Tip izraza dodele je tip leve strane, a vrednost izraza dodele je vrednost koja će biti dodeljena levoj strani
- Promena vrednosti objekta na levoj strani je *propratni (bočni, sporedni) efekat* (engl. side effect) do kojeg dolazi prilikom izračunavanja vrednosti izraza.

Operator dodele

- Na primer, izvršavanje naredbe `broj_studenata = 80;` svodi se na izračunavanje izraza `broj_studenata = 80`.
- Tip promenljive `broj_studenata` je istovremeno i tip ovog izraza, vrednost je jednaka 80, a prilikom ovog izračunavanja menja se vrednost promenljive `broj_studenata`.
- Ovakvo ponašanje može se iskoristiti i za višestruko dodeljivanje.
- Na primer, nakon sledeće naredbe, sve tri promenljive `x`, `y` i `z` imaće vrednost 0:

```
x = y = z = 0;
```

Aritmetički operatori

- `+` binarni operator sabiranja;
- `-` binarni operator oduzimanja;
- `*` binarni operator množenja;
- `/` binarni operator (celobrojnog) deljenja;
- `%` binarni operator ostatka pri deljenju;
- `-` unarni operator promene znaka;
- `+` unarni operator.

Svi navedeni binarni operatori imaju levu asocijativnost.
% se može primeniti samo na cele brojeve
/ se može primeniti i na cele i na realne brojeve

Inkrementiranje i dekrementiranje

- Operator inkrementiranja (uvećavanja za 1) zapisuje se sa ++, a operator dekrementiranja (umanjivanja za 1) zapisuje se sa --:
- Prefiksni i postfiksni

```
x = n++;
x = ++n;
```

```
int a = 3, x = a++, y = a++;
int b = 3, z = b++ + b++;
printf("a = %d, x = %d, y = %d,\n", a, x, y);
printf("b = %d, z = %d\n", b, z);
```

Relacijski operatori

- `>` veće;
- `>=` veće ili jednako;
- `<` manje;
- `<=` manje ili jednako;
- `==` jednako;
- `!=` različito.

Prioritet: prva četiri isti, viši od jednakosti i različitosti
Leva asocijativnost
Rezultat relacionog operatora primenjenog nad dva broja je vrednost 0 (koja odgovara istinitosnoj vrednosti *netačno*) ili vrednost 1 (koja odgovara istinitosnoj vrednosti *tačno*).
Binarni relacijski operatori imaju niži prioritet od binarnih aritmetičkih operatora.

Relacijski operatori

- `3 > 5` ima vrednost 0
- `7 < 5 != 1` ima vrednost 1
- Ako promenljiva `x` ima vrednost 2, izraz `3 < x < 5` ima vrednost 1
- Operator == koji ispituje da li su neke dve vrednosti jednake i operator dodele = različiti su operatori i imaju potpuno drugačiju semantiku. Njihovo nehotično mešanje čest je uzrok grešaka u C programima.

Logički operatori

`!` logička negacija

`&&` logička konjunkcija

`||` logička disjunkcija

Logički operatori primenjuju se nad brojevnim vrednostima i imaju tip rezultata `int`.
Brojevnim vrednostima pridružene su logičke ili istinitosne vrednosti na sledeći način: ukoliko je broj jednak 0, onda je njegova logička vrednost 0 (*netačno*), a inače je njegova logička vrednost 1 (*tačno*).

Logički operatori

- vrednost izraza `5 && 4.3` jednaka je `1`;
- vrednost izraza `10.2 || 0` jednaka je `1`;
- vrednost izraza `0 && 5` jednaka je `0`;
- vrednost izraza `!1` jednaka je `0`;
- vrednost izraza `!9.2` jednaka je `0`;
- vrednost izraza `!0` jednaka je `1`;
- vrednost izraza `!(2>3)` jednaka je `1`;
- izrazom `3 < x && x < 5` proverava se da li je vrednost promenljive `x` između 3 i 5;
- izraz `a > b || b > c && b > d` ekvivalentan je izrazu `(a>b) || ((b>c) && (b>d))`;
- izrazom `g % 4 == 0 && g % 100 != 0 || g % 400 == 0` proverava se da li je godina `g` prestupna.

Lenjo izračunavanje

- U izračunavanju vrednosti logičkih izraza koristi se strategija *lenjog izračunavanja* (engl. lazy evaluation).
- Osnovna karakteristika ove strategije je izračunavanje samo onog što je neophodno.
- ```
  2<1 && a++
  a++ && 2<1
  1<2 || a++
  2<1 || a++
  ```
  ```

Složeni operatori dodele

- Dodela koja uključuje aritmetički operator `i = i + 2;` može se zapisati kraće i kao `i += 2;`. Slično, naredba `x = x * (y+1);` ima isto dejstvo kao i `x *= y+1;`.
- Za većinu binarnih operatora postoje odgovarajući složeni operatori dodele: `+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `«=`, `»=`
- Operatori dodele imaju niži prioritet od svih ostalih operatora i desnu asocijativnost.

Operator uslova

Ternarni operator uslova `?:` se koristi u sledećem opštem obliku:

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

Prioritet ovog operatora je niži u odnosu na sve binarne operatore osim dodela i operatora `,`.

```
max = (a > b) ? a : b;
abs = (a < 0) ? -a : a;
n = 0;
x = (2 > 3) ? n++ : 9; /*lenja semantika*/
```

Operator , i sizeof

- Binarni operator zarez `,` je operator najnižeg prioriteta.
- Prilikom izračunavanja vrednosti izraza izgrađenog njegovom primenom, izračunavaju se oba operanda, pri čemu se vrednost celokupnog izraza definiše kao vrednost desnog operanda.

```
x = 3, y = 5; /* ekivalentno bi bilo i x = 3; y = 5; */
```

- Veličinu u bajtovima koju zauzima neki tip ili neka promenljiva moguće je odrediti korišćenjem operatora `sizeof`.
- Tako, `sizeof(int)` predstavlja veličinu tipa `int` i na tridesetidvobitnim sistemima vrednost ovog izraza je najčešće 4.

Pregled

Konverzije tipova

- Konverzija tipova predstavlja pretvaranje vrednosti jednog tipa u vrednost drugog tipa.
- Jezik C je veoma fleksibilan po pitanju konverzije tipova i u mnogim situacijama dopušta korišćenje vrednosti jednog tipa tamo gde se očekuje vrednost drugog tipa
- Vrste konverzija: eksplicitne i implicitne
- Neke konverzije je moguće izvesti bez gubitka informacija, dok se u nekim slučajevima prilikom konverzije vrši izmena same vrednosti podatka.

Konverzije tipova

- Jedan oblik konverzije predstavlja konverzija vrednosti „nižeg tipa“ u vrednost „višeg tipa“ (na primer, `short` u `long`, `int` u `float` ili `float` u `double`) u kom slučaju najčešće ne dolazi do gubitka informacije.
- Konverzija tog oblika se naziva *promocija* (ili *napredovanje*)

```
float a=4;  /* 4 je int i implicitno se konvertuje u flo
float f = 16777217; /*gubitak informacije*/
```

Konverzije tipova

- Drugi oblik konverzije predstavlja konverzija vrednosti višeg tipa u vrednost nižeg tipa (na primer, `long` u `short`, `double` u `int`).
- Ovaj oblik konverzije ponekad se naziva *democija* (ili *nazadovanje*).
- Prilikom ovog oblika konverzije, moguće je da dođe do gubitka informacije (u slučaju da se polazna vrednost ne može predstaviti u okviru novog tipa).

```
int b = 7.0f; /* 7.0f je float pa se vrsi konverzija u 7
int c = 7.7f; /* 7.7f je float i vrsi se konverzija u 7,
                 pri cemu se gubi informacija */
unsigned char d = 256; /* d dobija vrednost 0 */
```

Konverzije tipova

- Prilikom konverzije iz brojeva u pokretnom zarezu u celobrojne tipove podataka i obratno potrebno je potpuno izmeniti interni zapis podataka (na primer, iz IEEE754 zapisa brojeva u pokretnom zarezu u zapis u obliku potpunog komplementa).
- Ovo se najčešće vrši uz „odsecanje decimala“, a ne zaokruživanjem na najbliži ceo broj (tako je `7.7f` konvertovano u 7, a ne u 8).
- Prilikom konverzija celobrojnih tipova istog internog zapisa različite širine (različitog broja bajtova), vrši se odsecanje vodećih bitova zapisa (u slučaju konverzija u uži tip) ili proširivanje zapisa dodavanjem vodećih bitova (u slučaju konverzija u širi tip).

Konverzije tipova

- Eksplicitna i implicitna konverzija
- Eksplicitna konverzija: kada programer navede u koji tip želi da se konverzija izvrši
- Implicitna konverzija: kada kompajler sam odlučuje o konverziji iz jednog tipa u drugi

Eksplicitna konverzija

- `(tip)izraz`
- Operator kastovanja je prefiksni, unaran operator i ima viši prioritet od svih binarnih operatora.
- U slučaju primene operatora kastovanja na promenljivu, vrednost izraza je vrednost promenljive konvertovana u traženi tip, a vrednost same promenljive se ne menja (i, naravno, ne menja se njen tip).

```
int a = 13, b = 4;
printf("%d\t", a/b);
printf("%f\n", (double)a/(double)b);
```

Impicitna konverzija

- Prilikom primene nekih operatora vrše se konverzije vrednosti operanada implicitno, bez zahteva progremera.

```
int a;
double b = (a = 3.5);
```

- Prilikom primene nekih aritmetičkih operatora vrše se implicitne konverzije (najčešće promocije) koje obezbeđuju da operandi postanu istog tipa pogodnog za primenu operacija.
- Na primer,

```
int a = 3;
double b = 4.5;
a+b <- Tip ovog izraza je double
```

Celobrojna promocija

- Aritmetički operatori se ne primenjuju na „male“ tipove tj. na podatke tipa `char` i `short` (zbog toga što je u tim slučajevima verovatno da će doći do prekoračenja tj. da rezultat neće moći da se zapiše u okviru malog tipa), već se pre primene operatora mali tipovi promovišu u tip `int`.

```
unsigned char cresult, c1, c2, c3;
c1 = 100;
c2 = 3;
c3 = 4;
cresult = c1 * c2 / c3;
```

- Rezultat će biti ispravan, iako broj 300 ne može da stane u char

Implicitna konverzija

1. Ako je bar jedan od operanada tipa `long double`, onda se drugi konvertuje u `long double`;
2. inače, ako je jedan od operanada tipa `double`, onda se drugi konvertuje u `double`;
3. inače, ako je jedan od operanada tipa `float`, onda se drugi konvertuje u `float`;
4. inače, svi operandi tipa `char` i `short` promovišu se u `int`.
5. ako je jedan od operanada tipa `long long`, onda se drugi konvertuje u `long long`;
6. inače, ako je jedan od operanada tipa `long`, onda se drugi konvertuje u `long`.

Označeni i neoznačeni brojevi

- U slučaju korišćenja neoznačenih operanada (tj. mešanja označenih i neoznačenih operanada), pravila konverzije su nešto komplikovanija
- Ako je neki tip širi, onda se konverzija vrši u širi tip
- Ako su oba tipa iste širine, onda se konverzija vrši u neoznačeni tip
- Problemi tu nastaju prilikom poređenja označenih i neoznačanih tipova. Na primer, ne važi da je `-1l < 1ul`

Literatura

Slajdovi su pripremljeni na osnovu materijala iz petog i šestog poglavlja knjige:
Filip Marić, Predrag Janičić: Programiranje 1
Za pripremu ispita nisu dovoljni slajdovi, potrebno je koristiti knjigu!