Programski jezik I

Tipovi podataka i deklaracija promjenljivih

Tipovi podataka

- Sve promjenljive se u C-u moraju eksplicitno deklarisati prije prve upotrebe.
- Ovaj korak je neophodan kako bi omogućili zauzimanje memorije za promjenljive.
- Sve promjenljive pripadaju jednom od 10 tipova podataka.
- Osnovni (elementarni) tipovi podataka:
  - cijeli broj (int ili long),
  - realni broj (float ili double) i
  - karakter (char).

Tipovi podataka

- Složeni ili izvedeni tipovi podataka su:
  - neodređeni tip (void),
  - pokazivač (eng. pointer),
  - nabrajanje (enumeracija),
  - niz (ponekad se zove vektor; pod nizovima uključujemo i višedimenzione nizove, odnosno matrice, kao i stringove),
  - struktura (koriste se i pojmovi slog, record, zapis),
  - unija i
  - fajl.
- Pored ovoga, na osnovu struktura se mogu definisati i složeni linkovani tipovi podataka (liste, grafovi, stabla).
- Danas ćemo se upoznati sa osnovnim, a do kraja kursa i sa ostalim tipovima podataka.

Deklaracija promjenljivih

- Sve promjenljive u C programu se moraju deklarisati prije prvog korišćenja.
- Deklaracija podrazumijeva navođenje tipa i imena promjenljive.
- Ilustrujmo deklaraciju na primjeru cjelobrojnog tipa **int**.

```
int i;
```

Deklaracija (najava za korišćenje, odnosno zauzimanje prostor u memoriji) cjelobrojne promjenljive i. Ne znamo šta se trenutno nalazi upisano u toj promjenljivoj, jer to zavisi od prethodnog stanja u memoriji.

```
int i;
i=0;
```

Nakon deklaracije, negdje u programu moramo postaviti početnu vrijednost promjenljive. Ovo se naziva **inicijalizacija**.

Deklaracija i inicijalizacija

```
int i=0;
```
→ Deklaracija i inicijalizacija se često kombinuju kako bi izbjegli besmislene vrijednosti u promjenljivim. Ovo se naziva **definicijom** promjenljive.

```
int i=7, b=0, c;
```
→ Deklaracije i definicije se mogu kombinovati.

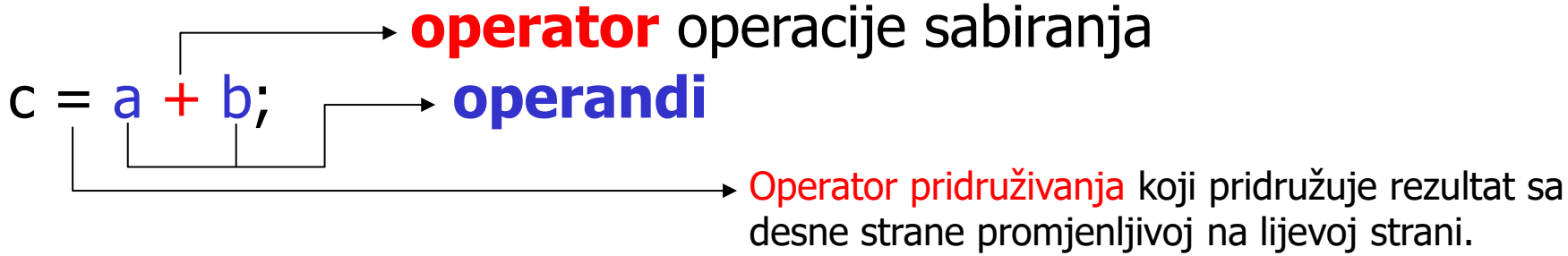
```
int i, j=i+2;
```
→ Dozvoljen, ali besmislen oblik deklaracije.

```
int j=i;
int i;
```
→ Nedozvoljeno! U trenutku inicijalizacije j ne postoji i.

Nije dozvoljeno unutar jednog **bloka** naredbi uvesti dva puta istu promjenljivu.

Blok naredbi je skup naredbi oivičen vitičastim zagradama **{ }**.

Operacije – operatori - operandi



- Oznake **5** osnovnih matematičkih operacija su:
  + (sabiranje i pozitivan predznak), **-** (oduzimanje i negativni predznak), ***** (množenje), **/** (dijeljenje) i **%** (ostatak pri dijeljenju – isključivo se primjenjuje kod cijelih brojeva).
- Prioritet operacija je: (1) predznak, (2) množenje, dijeljenje i ostatak pri dijeljenju i (3) sabiranje i oduzimanje.
- Kad god postoji potreba ili dilema, koriste se zagrade **()** za promjenu redosljeda izvršavanja operacija: a = (a+b)*2 + ((c-d%e)*(b-d))/2;

Skraćena notacija

- U programiranju su česte operacije tipa: **a=a+b;**
- Za operacije ovog tipa, u C-u su uvedene skraćene notacije:

  a += b; a -= b; a *= b; a /= b; a %= b;

  koje znače redom:

  a = a + b; a = a - b; a = a * b; a = a / b; a = a % b;
- Već ste (nadamo se) uočili karakter **;** koji stoji na kraju svake od naredbi i znači kraj naredbe.
- Ovdje se ne završava bogatstvo operatora koje je bilo nekarakteristično za programske jezike prije C-a.

Inkrementiranje i dekrementiranje

- U programskim jezicima se često izvršava naredba tipa **i=i+1** kojom se promjenljiva **i** uvećava za **1**. Ovo se naziva inkrementiranje.
- Za obavljanje ove operacije potrebno je nekoliko instrukcija:
  - smještanje **i** u registar računara;
  - generisanje konstante **1** u drugom registru;
  - sabiranje i privremeno memorisanje rezultata;
  - kopiranje rezultata iz registra u memoriju u promjenljivu **i**.
- Ako se podsjetite kursa OR, svi procesori, pa čak i korišćeni primitivni model procesora, imaju instrukciju koja inkrementira (povećava vrijednost za 1).

Inkrementiranje i dekrementiranje

- Stoga, C, kao jezik koji ima dobru komunikaciju sa hardverom, ima operator koji direktno poziva procesorsku instrukciju za inkrementiranje. Postoje 2 verzije operatora:

  k++ (postfiksna verzija: prvo upotrijebi k u izrazu, a zatim ga uvećaj za 1)
  ++k (prefiksna verzija: prvo povećaj za 1, a zatim ga upotrijebi u izrazu)

  Na primjer: k=0; j=k++; daje k=1 i j=0
  k=0; j=++k; daje k=1 i j=1

- Izraze u kojima više puta figuriše promjenljiva koja se inkrementira, npr. y = (x++) * x, treba izbjegavati jer rezultat može zavisiti od kompajlera (nedefinisan rezultat).
- Postoji i operator --, u dvije varijante, k-- i --k, koji služi za dekrementiranje (umanjivanje za 1). Isti princip kao kod ++.

Operatori operacija poređenja

- Operatori operacija poređenja su:
  - **>** (veće od)
  - **<** (manje od)
  - **>=** (veće od ili jednako)
  - **<=** (manje od ili jednako)
  - **!=** (provjera nejednakosti)
  - **==** (provjera jednakosti; uočite razliku u odnosu na operator pridruživanja – česta greška kod početnika!)

Aritmetičke operacije imaju veći prioritet od logičkih.

- Ako je poređenje zadovoljeno rezultat je 1, a ako nije 0.
- Kao logičku istinu programski jezik C tumači svaku vrijednost različitu od nule!

Operatori logičkih operacija

- Operatori logičkih operacija su:
  - **!** je operator negacije (od izraza različitog od 0 daje 0, i od 0 daje 1),
  - **&&** logička operacija AND (logičko I),
  - **||** logička operacija OR (logičko ILI).
- Dozvoljene su i skraćene verzije operatora tipa **&&=**.
  - Logički izraz 5||0 daje 1 jer se 5 tumači kao logička istina.
  - Kod upotrebe logičkih operatora u složenim izrazima oprez!

```
int k = 2, j;
j = (2 || k++);
```

Nakon ove dvije naredbe je j = 1 i k = 2, jer prilikom izvršavanja složenog logičkog izraza se prvo pogleda prvi dio izraza i, kako je on tačan, cio logički izraz je sigurno tačan (logičko ILI). Tako se, u ovom slučaju, uopšte ne izvršava izraz k++, pa k ostaje 2.

Ternarni operator

- U programskom jeziku C postoji ternarni (koji ima tri operanda) operator **?:**

  uslov ? izraz1 : izraz2

- Ako je logički uslov tačan, vraća se vrijednost izraza izraz1, a ako nije vrijednost izraz2.

- Primjer:

  a = (j>2) ? (j+2) : (j-2);

  Ako je j>2, izvršava se a=j+2, a ako nije a=j-2.

Operacije sa bitovima

- Programski jezik C posjeduje, kao snažan koncept, operacije za direktan rad sa bitovima, što se vrlo često koristi prilikom programiranja hardvera.
- Postoje dvije grupe operatora za rad na nivou bitova:
  - **logičke operacije nad bitovima:** **~** negacija po bitovima, npr. ~k znači gdje je u binarnom zapisu k bila 0 postavlja 1 i obratno; **a&b** logička operacija I nad bitovima (bit rezultata je 1 samo ako su biti na istom mjestu u a i b jednaki 1); **a|b** logičko ILI nad bitovima; **a^b** ekskluzivno ILI nad bitovima.
  - **operacije pomjeranja:** **k<<p** znači pomjeranje binarne reprezentacije broja k za p bita u lijevo, a na upražnjena mjesta se upisuju nule. Sa druge strane, k>>p je pomjeranje broja k za p bita u desno. Posljedica: k<<1 odgovara množenju broja k sa 2, k>>1 dijeljenju sa 2.

Operator koma ,

- Posljednji operator koji će za sada biti uveden je operator **koma** (odnosno zarez) koji se koristi za nabrajanje izraza, a rezultat operacije je vrijednost izraza koji je krajnje desno. Na primjer, i = (j=2, 3) će postaviti j na 2, a zatim će 3 pridružiti promjenljivoj i.
- Sada prelazimo na osnovne tipove podataka.

Tip int i modifikacije

- Ispred riječi int, mogu se naći ključne riječi **short** i **long**.
- Ako se promjenljiva deklariše kao **short int** (dovoljno je pisati samo short), zauzeće se manje memorije, što za posljedicu ima manji opseg brojeva sa kojim se radi.
- Za "dugačke" cijele brojeve, koji zauzimaju više prostora u memoriji, koriste se deklaracije **long int** (dozvoljeno je pisati samo long) i **long long int** (dozvoljeno je samo long long).
- Postoje i modifikatori **signed** (podrazumijeva se) i **unsigned** koji se mogu kombinovati sa short i long.
- **unsigned** znači da sadržaj memorije treba tumačiti kao cijeli broj koji ne može uzeti negativnu vrijednost (npr. umjesto da se brojevi tumače u domenu od -2^{15} do $2^{15}-1$ oni se tumače kao brojevi u domenu 0 do $2^{16}-1$).

Tipovi float i double

- Realni brojevi se deklarišu kao **float** (kratki memorijski zapis) ili kao **double** (dugi memorijski zapis).
- Kod modernih računara, binarna reprezentacija realnih brojeva u pokretnim zarezu je u skladu sa IEEE 754 standardom, po kome se realni brojevi opisuju sa tri cijela broja: **znakom s** (jedan bit), **mantisom c** (ili koeficijentom) i **eksponentom q**. Broj ima vrijednost

$$(-1)^s \times c \times 2^q$$

- Opseg vrijednosti tipova float i double je dat u tabeli nekoliko slajdova naprijed.

Tip char

- Tip podataka **char** predstavlja karakter i zapisuje se u memoriji kao cijeli broj koji zauzima **1 bajt** i koji se tumači na osnovu **ASCII tabele**. Ovako deklarisani podaci su zapravo cijeli brojevi u intervalu –128 do 127. Nama su interesantni samo pozitivni brojevi.
- Domen promjenljivih tipa **unsigned char** je 0 - 255.
- Deklaracija promjenljive a tipa char i dodjela konkretnog karaktera se vrši na sljedeći način: char a = '#'.
- Numerička vrijednost karaktera odgovara njegovoj poziciji u ASCII tabeli. Tako, na primjer, karakteru '+' odgovara broj 43, slovu 'T' broj 84, slovu 't' broj 116, cifri '7' odgovara broj 55.

Pogodne osobine ASCII tabele

- ASCII tabela ima nekoliko pogodnih osobina:
  - mala slova su poređana jedno za drugim po engleskom alfabetu (od 'a' do 'z'),
  - velika slova su poređana jedno za drugim po engleskom alfabetu (od 'A' do 'Z'),
  - cifre su poređane jedna za drugom (od '0' do '9').
- Stoga nam nije bitan međusoban odnos malih i velikih slova i cifara, kao ni ASCII kôd pojedinih karaktera.

Operacije kod karaktera

- Zbog prethodno opisanih osobina ASCII tabele, nad karakterima se mogu provoditi sljedeće operacije:
  - Aritmetičke operacije. Na primjer, sabiranje **A='a'+1** daje **A='b'**. Zbog čega? Naravno, neke operacije nemaju previše smisla, ali nijesu zabranjene, npr. B = 'a' * '0'.
  - Operacije poređenja. Ima smisla porediti samo karaktere koji pripadaju istoj grupi (međusobno poređenje malih ili velikih slova), ali nije zabranjeno, ali ni previše smisleno, poređenje karaktera iz različitih grupa.

Elementarni tipovi podataka

Tip promjenljive	Broj bajtova	Opseg vrijednosti
char	1	-128 ÷ 127
unsigned char	1	0 ÷ 255
short int	2	-32 768 ÷ 32 767
unsigned short int	2	0 ÷ 65 535
int	2 ili 4	-32 768 ÷ 32 767 ili -2 147 483 648 ÷ 2 147 483 647
unsigned int	2	0 ÷ 65 535
long int	4	-2 147 483 648 ÷ 2 147 483 647
unsigned long int	4	0 ÷ 4 294 967 295
long long int	8	-9 223 372 036 854 775 808 ÷ 9 223 372 036 854 775 807
unsigned long long int	8	0 ÷ 18 446 744 073 709 551 615
float	4	$-3{,}4028\times10^{+38} \div 3{,}4028\times10^{+38}$
double	8	$-1{,}7977\times10^{+308} \div 1{,}7977\times10^{+308}$
long double	10, 12 ili 16	---

Literali za numeričke tipove

```
int x = 22;              // 22 je literal (konstanta) tipa int
long x = 22L;            // 22L ili 22l su literali tipa long
long long x = 22LL;      // 22LL ili 22ll su literali tipa long long
```

Konstanta 12345678912 je takođe tipa long long jer ta vrijednost ne može da stane u tip long (4 bajta)

```
float x = 2.3F;          // 2.3F ili 2.3f su literali tipa float
double x = 2.3;          // 2.3 je literal tipa double
long double x = 2.3L;    // 2.3L ili 2.3l su literali tipa long double
```

```
float x = 2F;      // Greška! Sufiks F se ne dodaje
                   // na cjelobrojnu konstantu.
```

Konverzija podataka

- Prirodno je očekivati da u operacijama učestvuju podaci različitih tipova. U programskom jeziku C postoje specifična pravila koja određuju ponašanje u tom slučaju.
- Da bismo ih objasnili, uvedimo cjelobrojne promjenljive intA i intB, i realne promjenljive floatA i floatB. Neka su njihove vrijednosti: intA = 4, intB = 3, floatA = 2.1, floatB = 1.7.
- Moguće situacije:

1) floatA=intA+intB daje floatA=7, što je očekivano.
2) floatA=intA/intB daje floatA=1, jer je za rezultat ostavljeno mjesta kao za cijeli broj (oba operanda su cijeli brojevi) i dolazi do odsijecanja necjelobrojnog dijela.

Konverzija podataka

- Moguće situacije (intA=4, intB=3, floatA=2.1, floatB=1.7):

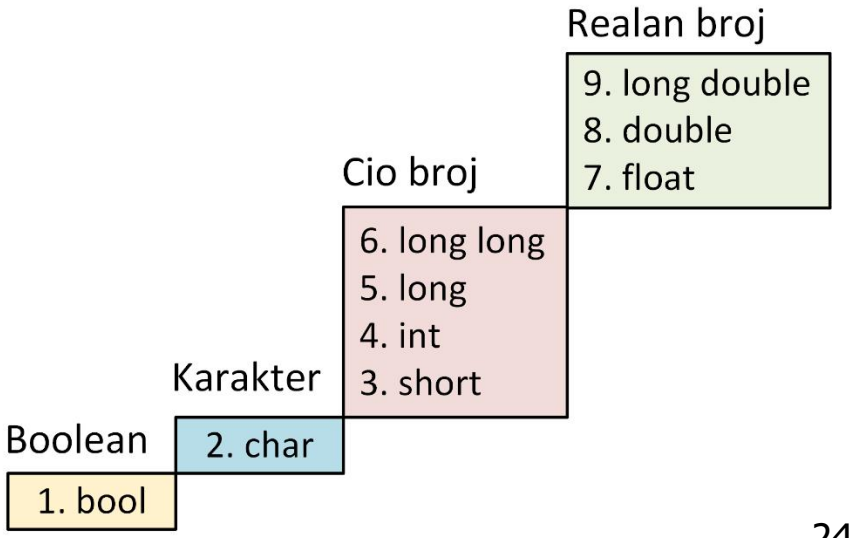
3) intA=floatA+floatB daje rezultat intA=3. Obavi se sabiranje za realne brojeve, ali prilikom prebacivanja rezultata u cjelobrojnu promjenljivu dolazi do odsijecanja necjelobrojnog dijela.

4) floatB=floatA+intA daje floatB=6.1. Tip rezultata je isti kao tip "složenijeg" od dva operanda, u ovom slučaju float.

5) intB=floatA+intA daje intB=6. Sami protumačite zašto.

Implicitna konverzija podataka

- U operacijama sa različitim tipovima operanada, vrši se implicitna konverzija podataka => vrijednost nižeg tipa (npr. int vrijednost 2) se konvertuje u odgovarajuću vrijednost višeg tipa (npr. double vrijednost 2.0).
- Implicitna konverzija se obavlja nezavisno od korisnika.
- Pravila za konverziju su data na slici desno:



Konverzija prilikom poziva funkcije

- Vratite se na ovaj slajd kad budemo obrađivali funkcije ☺
- Argumenti funkcija u C-u moraju imati najavljene tipove.
- Ako se proslijedi argument koji je različitog tipa od onog koji se očekuje doći će do konverzije podataka.
- Ako se očekuje argument koji je "većeg" tipa nego onaj koji je proslijeđen, proslijeđeni podatak se konvertuje u "veći" tip.
- Ako se očekuje argument nekog tipa, a bude proslijeđen argument "većeg" tipa, prilikom korišćenja u funkciji koristi se samo dio informacija iz argumenta.
- U slučaju nesaglasnosti tipova argumenata kod funkcija može doći do čudnih grešaka u programu, jer je C relativno nekorektan i pokušava da izvrši svaku dozvoljenu konverziju podataka.

Eksplicitna konverzija

- Programski jezik C dozvoljava korisniku da sam podešava konverziju podataka.
- Korisnička konverzija podataka se naziva eksplicitnom.
- Preporučuje se korisniku da kad god je potrebno izvrši eksplicitnu konverziju!
- Za eksplicitnu konverziju se koristi cast operator. Oblik ovog operatora je:

  (tip_promjenljive) promjenljiva → cast operator

- U operaciji gdje se koristi ovakav iskaz promjenljiva će se tretirati kao da je tipa tip_promjenljive.

Eksplicitna konverzija - primjer

- Na primjer, neka je floatA realna promjenljiva (tipa float) i neka su intA=5 i intB=2 cjelobrojne promjenljive. Operacija

  floatA = (float) intA / intB

  daje "korektan" rezultat floatA=2.5, jer se sada vrijednost promjenljive intA tretira kao da je tipa float.
- Napomena: Promjenljiva intA nije promijenila tip u float, već se u izrazu koristi realni broj čija je vrijednost ista kao vrijednost cjelobrojne promjenljive intA.
- Rezultat operacije (int)3.4+5 je 8, jer se vrši tretiranje prvog argumenta kao da je u pitanju cjelobrojna promjenljiva.
- Preporučujemo korišćenje ekplicitne konverzije kad god može doći do konverzije podataka.

Operator sizeof

- Vidjeli smo da su neke veličine, kao što je memorija koju zauzimaju promjenljive, mašinski zavisne.
- Programe treba pisati tako da se izbjegne mašinska zavisnost.
- Jedan od elemenata koji tu pomaže je operator **sizeof** **(liči na funkciju ili naredbu, ali je, zapravo, operator)** koji vraća broj bajtova koji neki memorijski objekat (promjenljiva) zauzima. Može da odredi koliko memorije zauzima i tip podatka.
- Primjeri: sizeof(long), sizeof(i), sizeof(60).
- Dozvoljen je i zapis bez zagrada za operand, sa izuzetkom tipa podatka, tj. dozvoljeno je sizeof k, ali sne i sizeof int.