Programiranje I

Deklaracija promjenljivih i tipovi podataka

Tipovi podataka

- Sve promjenljive se u C-u moraju eksplicitno deklarisati prije 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),
  - realni broj (float ili double) i
  - karakter (char).

Tipovi podataka

- Složeni ili izvedeni tipovi podataka su:
  - neodređeni tip (void),
  - pokazivač (pointer),
  - nabrajanje (enumeracija),
  - niz (ponekad se zove kolekcija ili vektor; pod nizovima uključujemo i višedimenzione nizove, odnosno matrice, kao i stringove),
  - struktura (struct, a 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.

Tip int

- Sve promjenljive se deklarišu prije bilo koje druge naredbe u programu.
- Cijeli brojevi se najavljuju pomoću ključne riječi **int**.
- Postoje i varijante da se zatraži zauzimanje manje memorije za cijeli broj. Tada se promjenljiva deklariše kao **short int** **(dovoljno je postaviti short jer se int podrazumjeva).** Manje memorije istovremeno podrazumijeva i manji opseg brojeva koji se može dodijeliti datoj promjenljivoj.
- Za "dugačke" cijele brojeve, koji zauzimaju više prostora u memoriji, koristi se deklaracija **long int** **(opet je dozvoljeno samo long).**

Tip int i modifikacije

- Koliko je to manje ili više nije stvar programskog jezika C već kompajlera, hardvera i operativnog sistema.
- Pored ovoga mogu se dodati i modifikatori **signed** **(može se uvijek izostaviti)** 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 $2^{16}-1$).

Deklaracija i inicijalizacija

```
int i;
```

Deklaracija (najava za korišćenje, odnosno zauzimanje prostora u memoriji) cjelobrojne promjenljive i. Ne znamo što 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 neku početnu vrijednost promjenljive. Ovo se naziva **inicijalizacija**.

```
int i=0;
```

Programeri često vrše deklaraciju i inicijalizaciju zajedno kako bi izbjegli da im ijednog trenutka u promjenljivim budu upisane "besmislene" vrijednosti. Ovo se naziva **definicijom** promjenljive.

```
int i=7, b=0, c;
```

Deklaracije i definicije se mogu kombinovati.

Deklaracija promjenljivih

```
int i;
int j=i+2;
```

```
int i, j=i+2;
```

Dozvoljeni, ali besmisleni oblici deklaracije. Razumno je:

```
int i=0;
int j=i+2;
```

```
int j=i;
int i;
```

Nedozvoljen oblik deklaracije, jer u trenutku inicijalizacije promjenljive j ne postoji promjenljiva i.

Nije dozvoljeno unutar istog bloka naredbi uvesti dva puta isto ime, tj. dva puta istu promjenljivu.

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

Operator sizeof

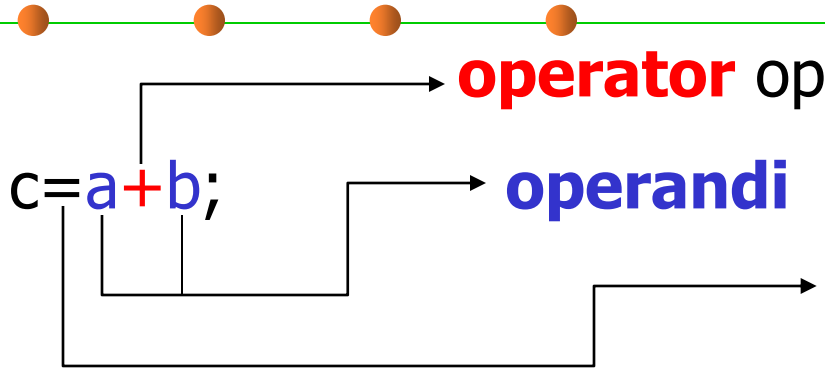
- Vidjeli smo da su neke veličine, kao što je memorija koju zauzimaju promjenljive, mašinski zavisne.
- Programe treba pisati tako da se izbjegne mašinska zavisnost.
- Jedan od elemenata koji tu pomaže je operator **sizeof** **(liči na funkciju ili naredbu, ali je, zapravo, operator)** koji vraća broj bajtova koji neki memorijski objekat (promjenljiva) zauzima. Može da izmjeri i koliko memorije zauzima tip podataka. Na primjer, **sizeof(int)**, **sizeof(long)**, **sizeof(i)**, **sizeof(60)**, a može čak i zapis bez zagrada za operand, osim kada je operand tip podataka, tj. dozvoljeno je **sizeof i**, ali ne i **sizeof int**.

Operacije – operatori - operandi

- U C-u možemo saznati memorijsku lokaciju promjenljive pomoću operatora **&**, npr. **&i**. Ovo je unarni operator jer se primijenjuje nad jednom promjenljivom. Operacija se može primijeniti na sve tipove podataka.
- Oznake **5** osnovnih matematičkih operacija su:
  + (sabiranje i pozitivan predznak, ali se kao predznak rijetko koristi),
  - (oduzimanje i negativni predznak),
  ***** (množenje),
  / (dijeljenje) i
  % (ostatak pri dijeljenju – isključivo se primjenjuje kod cijelih brojeva).

predznak je unarni operator, dok su ostali binarni (primjenjuju se nad dva operanda)

Operacije – operatori - operandi



Ako ste zaboravili, ovo nije matematičko jednako već operator pridruživanja koji pridružuje rezultat sa desne strane onome što se nalazi na lijevoj strani znaka jednakosti.

- Prioritet operacija je kao u matematici: predznak, ostatak pri dijeljenju, množenje, dijeljenje, i na kraju sabiranje i oduzimanje.
- Kad god postoji potreba ili dilema koriste se zagrade **()** za promjenu redosljeda izvršavanja operacija. Na primjer:
  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 znak **;** 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:
  - smiještanje **i** u registar računara;
  - generisanje konstante **1** u drugom registru;
  - sabiranje i privremeno memorisanje rezultata;
  - kopiranje rezultata iz registra u memoriju na promjenljivu **i**.
- Ako se podsjetite kursa OR1, 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 varijante operatora:

  k++ **(postfiksna varijanta koja znači prvo upotrijebi k u izrazu gdje se nalazi, a zatim ga uvećaj za 1) i**

  ++k **(prefiksna varijanta koja znači prvo povećaj k za 1, a zatim ga upotrijebi u izrazu).**

  Na primjer, k=0; j=k++; daje k=1 i j=0, dok k=0; j=++k; daje k=1 i j=1.

- Postoji operator --, opet u dvije varijante, k-- i --k, koji služi za dekrementiranje (umanjivanje za 1) sa istim principom kao kod inkrementiranja.

Operatori operacija poređenja

- Operatori operacija poređenja su:
  - **>** (veće od),
  - **<** (manje od),
  - **>=** (veće od ili jednako),
  - **<=** (manje od ili jednako),
  - **!=** (nije jednako; uočite razliku u odnosu na MATLAB) i
  - **==** (jednako; uočite razliku u odnosu na operator pridruživanja).
- Ako je poređenje zadovoljeno rezultat je 1, a ako nije 0.
- Kao logičku istinu programski jezik C tumači svaku vrijednost različitu od nule!

Aritmetičke operacije imaju veći prioritet od logičkih.

Operatori logičkih operacija

- Operatori logičkih operacija u C-u su:
  - **!** je operator negacije (od izraza različitog od 0 daje 0, i od 0 daje 1),
  - **&&** logička operacija AND – I,
  - **||** logička operacija OR – 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 i=2, j;
j = (2 || i++);
```

Nakon ove dvije naredbe je j=1 i i=2, jer prilikom izvršavanja naredbi se u ovom slučaju prvo pogleda prvi dio logičkog izraza i kako je on ispunjen logički izraz je sigurno tačan (operacija ILI je tačna kad je bilo koji od operanada tačan). Tako se u ovom slučaju uopšte ne izvršava drugi dio operacije, tj. i++ se ne izvršava, pa i ostaje i=2.

Ternarni operator

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

  uslov ? izraz1 : izraz2

- Ako je logički **uslov** tačan izvršava se **izraz1**, a ako nije **izraz2**. Na 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 je značajno u brojnim slučajevima pristupa hardveru.
- Postoje dvije grupe operatora za rad na nivou bitova:
  - **logičke operacije nad bitovima:** **~** negacija po bitovima, npr. **~i** znači gdje je u binarnom zapisu **i** bila nula postavlja jedan i obratno, **a&b** logička operacija I nad bitovima (rezultat ima 1 na onim mjestima gdje je 1 upisano i u a i u b), **a|b** logičko ILI nad bitovima, **a^b** ekskluzivno ILI nad bitovima.
  - **operacije pomjeranja:** **i<<k** znači pomjeranje i za k bita s desna, a na upražnjena mjesta se upisuju nule, i>>k je pomjeranje i s lijeva za k mjesta.
  - Za sve operatore koji rade sa bitovima osim unarnog (negacije po bitovima) mogu se uvesti skraćene oznake tipa **&=**.

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 realni broj kao tip podataka.
- Gotovo sve operacije koje se obavljaju nad cijelim brojevima se i ovdje mogu primijeniti (problem postoji kod ostatka, poređenja tipa **a==1** zbog načina zapisa realnih brojeva, kao i kod operacija sa bitovima, inkrementiranja, dekrementiranja, a postoji i niz drugih specifičnosti).

Tip podataka **float**

- 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 brojevi opisuju sa tri cijela broja: **znakom s** (jedan bit), **mantisom c** (ili koeficijentom) i **eksponentom q**. Broj ima vrijednost

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

- Ako se operacije izvršavaju nad podacima različitim po tipu izvršava se konverzija tipa podataka, koja će biti objašnjena nakon što bude objašnjen tip podataka **char**.

Tip podataka **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**.
- Ako deklarišemo promjenljive kao **unsigned char** domen je od **0** do **255**.
- Ako deklarišemo promjenljivu tipa **char** i dodjeljujemo joj konstantan karakter to moramo uraditi navođenjem unutar apostrofa:

```
char a;
a='c';
```

Podsjetite se i ostalih varijanti preko oktalnog ASCII koda, heksadecimalnog koda, kao i zadavanja specijalnih simbola.

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	-3,4028e+38 ÷ 3,4028e+38
double	8	-1,7977e+308 ÷ 1,7977e+308

“Zgodne” osobine ASCII tabele

- ASCII zapis (kao i drugi zapisi koji su u upotrebi) ima nekoliko zgodnih 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 kod pojedinih karaktera.

Operacije kod karaktera

- Zbog prethodno opisanih "zgodnih" osobina, nad karakterima se mogu provoditi, na prvi pogled, neočekivane operacije **(što slučaj u mnogim programskim jezicima)**:
  - Aritmetičke operacije. Na primjer, sabiranje **A='a'+1;** daje **A='b'**. Zbog čega? Naravno, neke operacije nemaju baš previše smisla, ali nijesu zabranjene, npr. B = 'a' * '0'.
  - Operacije poređenja. Naravno, ima smisla porediti samo karaktere koji pripadaju istoj grupi (međusobno poređenje malih slova), ali nije zabranjeno, ali ni previše smisleno, poređenje karaktera iz različitih grupa.

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 bi ih objasnili, uvedimo 4 promjenljive:
  - cjelobrojne iA i iB **(pojedine firme forsiraju svoje programere da koriste neka od pravila koja određuju kod imenovanja, kao što je ovdje uzeto da sa i počinju cjelobrojne, a sa f realne promjenljive),**
  - realne promjenljive fA i fB.
- Neka su vrijednosti iA=4, iB=3, fA=2.1 i fB=1.7.

Konverzija podataka

- fA=iA+iB daje fA=7, što je očekivano.
- fA=iA/iB daje fA=1, jer je za rezultat ostavljeno mjesta kao za cijeli broj (jer su oba operanda cijeli brojevi) i dolazi do odsjecanja necjelobrojnog dijela.
- iA=fA+fB daje rezultat iA=3. Obavi operaciju kao za realne brojeve, ali prilikom prebacivanja rezultata u cjelobrojnu promjenljivu dolazi do odsjecanja necjelobrojnog dijela.

Konverzija podataka

- fB=fA+iA daje fB=6.1. Računar "podesi" da je rezultat "komplikovanijeg" tipa od dva različita tipa operanada. To je u ovom slučaju tip podataka float.
- iB=fA+iA daje iB=6. Sami protumačite zbog čega.

- U operacijama u kojima učestvuju operandi različitih tipova prilikom smještanja promjenljivih u registar vrši se implicitna konverzija podataka, odnosno već prilikom kopiranja promjenljivih u registrima se podešava prostor za sve njih u skladu sa "najzahtijevnijom" promjenljivom koja učestvuje u operaciji.

Implicitna konverzija

- Implicitna konverzija je ona koja se na osnovu određenih pravila obavlja nezavisno od korisnika.
- Pravila za konverziju kod operacija u kojima učestvuju operandi različitog tipa su:
  - Vrijednosti tipa char, unsigned char, signed char i short int se pretvaraju u int.
  - Vrijednosti tipa unsigned short se pretvaraju u unsigned int.
  - Tip podatka koji zauzima manji memorijski prostor se pretvara u onaj koji zauzima veći. Na primjer, prilikom operacija u kojima učestvuju cijeli i realni brojevi dolazi do konverzije promjenljive tipa int u tip float.

Konverzija prilikom poziva 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.
- O ovome više riječi kada se budu učile funkcije.

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 fA realna promjenljiva (tipa float) i neka su iA=5 i iB=2 cjelobrojne promjenljive. Sada operacija: fA=(float)iA/iB daje "korektan" rezultat fA=2.5, jer se sada promjenljiva iA tretira kao da je tipa float.
- Napomena: Promjenljiva iA nije promjenila tip u prethodnom izrazu, već procesor koristi realni broj koji ima vrijednost cjelobrojne promjenljive iA.
- Ako je fB=3.4 i tipa float tada rezultat operacije: fA=(int)fB+iA je fA=8, jer se vrši tretiranje prvog argumenta kao da je u pitanju cjelobrojna promjenljiva.
- Preporučujemo vam korišćenje ekplicitne konverzije kad god može doći do konverzije podataka.