dr Bojana Dimić Surla

dr Dragan Urošević

UVOD U PROGRAMIRANJE

sa primerima u programskom jeziku C

Beograd, 2020.

Računarski fakultet

CET

Uvod u programiranje sa primerima u programskom jeziku C
dr Bojana Dimić Surla
dr Dragan Urošević

ISBN 978-86-7991-428-6

Recenzija	dr Danijela Boberić Krstićev, vanredni profesor Univerzitet u Novom Sadu, Prirodno-matematički fakultet
	dr Dragan Ivanović, vanredni profesor Univerzitet u Novom Sadu, Fakultet tehničkih nauka
Lektura	Jovana Janjić
Urednik	Jovana Ristić
Tehnički urednik	Vesna Petrinović
Izdavač	**CET Computer Equipment and Trade** Beograd, Skadarska 45 tel/fax: 011 3243-043, 3235-139, 3237-246 http://www.cet.rs
	Računarski fakultet Beograd, Knez Mihailova 6/VI tel: 011 2627-613, 2633-321 www.raf.edu.rs
Za izdavača	Dragan Stojanović, direktor
Tiraž	1000
Štampa	Pekograf, Beograd

Nastavno-naučno veće Računarskog fakulteta na 143. sednici održanoj 24. 6. 2020. godine donelo je odluku da se knjiga „Uvod u programiranje sa primerima u programskom jeziku C", autora dr Bojane Dimić Surla i dr Dragana Uroševića, bude štampana kao univerzitetski udžbenik.

Sadržaj

Predgovor

Ova knjiga je pisana kao udžbenik za predmet *Uvod u programiranje* koji se sluša u prvom semestru osnovnih akademskih studija studijskih programa Računarske nauke i Računarsko inženjerstvo na Računarskom fakultetu u Beogradu. Pored studenata Računarskog fakulteta, knjiga može biti od koristi svim početnicima koji ulaze u svet programiranja i žele da steknu temeljnu osnovu i suštinsko razumevanje ključnih koncepata programiranja.

Glavni fokus knjige je objašnjenje bitnih elemenata programiranja, načina razmišljanja koji stoji iza programa i algoritama, bez ulaženje u detalje o specifičnostima programskog jezika ili verzije kompajlera. Za prikaz primera korišćen je programski jezik C, ali se slična logika može koristiti u programiranju na drugim programskim jezicima.

Iako ne spada u moderne jezike, programski jezik C je čest izbor za početak učenja programiranja na mnogim svetskim i domaćim akademskim studijskim programima. Na ovakav izbor najviše utiče to što je sam jezik sa jedne strane dovoljno visokog nivoa da ne radimo sa mašinskim instrukcijama, a sa druge strane podržava koncepte programiranja nižeg nivoa kao što je upravljanje memorijom i rad sa pokazivačima. Sintaksa programskog jezika C je slična sintaksi savremenih viših programskih jezika, jer je većina tih jezika nastala upravo iz programskog jezika C, tako da privikavanje na sintaksu drugih jezika nakon učenja C-a najčešće ne zahteva veliki napor.

U prvom poglavlju objašnjeni su neki osnovni pojmovi iz programiranja kao što su programski jezik, kompajler, razvojno okruženje, predstavljanje podataka u računaru. Naredna poglavlja opisuju redom sve ključne koncepte programiranja od jednostavnijih ka složenijim, tako da se svako sledeće poglavlje nadovezuje na prethodno i koristi prethodno objašnjene elemente. U svakom poglavlju dat je veliki broj ilustrativnih primera i urađenih vežbi koji će čitaocu pomoći da bolje razume različite aspekte i principe teme iz progamiranja koja se obrađuje u poglavlju. Na kraju svakog poglavlja dati su dodatni zadaci za vežbu, manje i veće težine na kojima čitalac može da proveri svoje znanje.

Sa ciljem da se čitalac bolje pripremi za kasnije učenje programiranja i korišćenje literature na engleskom jeziku za veliki broj pojmova dat je i odgovarajući naziv na engleskom jeziku.

Kao što je ranije navedeno, izbor programskog jezika C omogućava da se obradi i koncept pokazivača koji ima značajnu ulogu u programiranju, iako ga većina viših programskih jezika ne podržava eksplicitno. Kroz učenje i razumevanje koncepta pokazivača stvara se dobra osnova za kasnije bolje razumevanje drugih paradigmi programiranja, kao na primer objektno-orijentisano programiranje. Sa druge strane, ovom knjigom obuhvaćeni su osnovi posle kojih čitalac može da se usmeri na različite oblasti računarstva, uključujući i programiranje nižeg nivoa (na primer programiranje operativnih sistema, sistema u realnom vremenu) za koje je neophodno poznavanje koncepta pokazivača.

Svi primeri programa u ovoj knjizi pisani su za komandnu liniju. Svi programski kodovi koji su dati na listinzima testirani su na 64-obitnom računaru, na operativnom sistemu Linux verzija Ubuntu 18.04.3 LTS sa ugrađenim C kompajlerom verzija gcc 7.5.0. Svi kodovi su javno dostupni i mogu se preuziti sa adrese https://github.com/RAFSoftLab/uup-udzbenik.

Zahvljujemo se recenzentima dr Danijeli Boberić Krstićev i dr Draganu Ivanoviću koji su svojim sugestijama doprineli kvalitetu sadržaja knjige, kao i svim dosadašnjim saradnicima na predmetu Uvod u programiranje koji su svojim idejama doprineli kvalitetu primera u ovoj knjizi.

Autori

Poglavlje 1

Programiranje i programski jezici

Računarstvo je ljudska delatnost koja se bavi rešavanjem problema korišćenjem računara. Iako se danas pod pojmom računarstva podrazumevaju i druge stvari, kao što je kreiranje kompjuterskih igrica, u svojoj osnovi ono jeste veoma efikasno sredstvo za rešavanje praktičnih problema ljudi. Programiranje, kao jedan veliki deo računarstva jeste pisanje kompjuterskih programa kojima se rešavaju neki zadati problemi.

Posmatrajmo sledeći problem:

`Napisati program koji na osnovu unetih dimenzija pravougaonika računa i ispisuje njegovu površinu.`

Ovakvi programi najčešće imaju tri dela, to su deo za učitavanje podataka, u kom od korisnika očekujemo da unese neke podatke koje će program da pročita (dimenzije pravougaonika), zatim deo za obradu podataka koji izvršava neke operacije nad učitanim podacima (množenje visine i širine pravougaonika) i deo za ispis podataka u kom se korisnik obaveštava o rezultatu izračunavanja (ispis izračunate površine). Osnovna logika programa sadržana je u delu za obradu podataka.

Niz tačno definisanih koraka kojim se rešava neki problem od strane računara naziva se *algoritam*. Ti koraci se nakada nazivaju i *komande* ili *naredbe*. Algoritam se može predstaviti grafički, posebnim neformalnim jezikom, tzv. pseudo kôdom ili se može opisati nekim slobodnim tekstom, a da bi algoritam bio razumljiv računaru mora se napisati u nekom *programskom jeziku* i u tom slučaju govorimo o *implementaciji* programa.

Tabela 1.1: Poređenje programa u višem programskom jeziku i mašinskom jeziku

Primer programa u mašinskom jeziku

```
0x A4 00 00 00
0x 60 01 00 84
0x A4 01 01 00
0x 60 02 00 00
0x 60 03 00 04
0x 60 04 00 00
0x 60 05 00 01
0x 08 00 00 02
0x 20 00 00 03
0x 20 04 04 05
0x 11 20 04 01
```

Primer programa u višem programskom jeziku

```
int main()
{
   int n;
   printf("Unesite broj\n");
   scanf("%d", &n);
   if (n%2 == 0)
      printf("Paran\n");
   else
      printf("Neparan\n");
   return 0;
}
```

1.1 Programski jezici

Postoji veliki broj programskih jezika koje se mogu prema različitim kriterijumima i karakteristikama podeliti u različite kategorije, na primer funkcionalni, proceduralni ili deklaritivni. Programski jezici dalje mogu biti niskog nivoa, što znači "bliži" računaru, na primer mašinski jezici, asembler. Programi pisani u ovakvim jezicima sadrže komande koje računarski procesor prepoznaje i izvršava. Karakteristika nižih programskih jezika je da su teže čitljivi i da je pisanje čak i vrlo jednostavnih programa veoma naporno, jer se svaka, i najmanja aktivnost mora napisati kao posebna naredba (na primer prelazak na određenu memorijsku lokaciju).

Jezici višeg nivoa su više prilagođeni čoveku, apstraktniji su, imaju više prepoznatljivih reči (najčešće iz engleskog jezika) i dosta funkcionalnosti je ugrađeno, ne moramo ručno da programiramo (na primer oslobađanje nepotrebne memorije). Primeri viših programskih jezika su Java, C, Python, C++, Visual Basic. Danas se programiranje po pravilu uvek uči na višem programskom jeziku, a i većina projekata u računarstvu se radi korišćenjem viših programskih jezika.

U tabeli 1.1 prikazani su primeri programa u mašinskom jeziku i višem programskom jeziku. Možemo primetiti da je program napisan u mašinskom jeziku nerazumljiv čoveku, dok u programu u višem jeziku možemo da prepoznamo neke reči engleskog jezika i nekakvu pravilnost u ispisu, na primer da se veliki broj redova završava oznakom tačka-zapeta (;). I mašinski jezici propisuju neke pravilnosti za pisanje programa i imaju naredbe, međutim programe na mašinskom jeziku vrlo retko, ili gotovo nikad ne piše čovek.

Pisanje programa predstavlja pisanje niza naredbi u određenom programskom jeziku, i taj niz naredbi naziva se *izvorni kôd programa* (eng. source code). Da bi se izvorni kôd programa napisan u nekom višem programskom jeziku mogao pokrenuti na računaru, on mora da se prevede na niži programski jezik, odnosno na mašinski jezik. Ovaj proces prevođenja programa napisanog u višem programskom jeziku u mašinski jezik se u nekim programskim jezicima naziva *kompajliranje*, dok se kod nekih jezika radi o *interpretiranju*. Kompajliranje se vrši korišćenjem programa koji se nazivaju *kompajleri*, dok se interpretiranje radi preko takozvanih *interpretera*. Kompajleri pretvaraju ceo program napisan u nekom višem programskom jeziku u mašinski jezik, tj. kôd koji računar zna da pročita i izvršava, posle čega taj program može da se izvršava. Kod jezika koji se interpretiraju, ne prevodi se ceo program u mašinski pre izvršavanja već se program izvršava tako što se jedna po jedna naredba prevodi u mašinski jezik u toku izvršavanja. Programi napisani na jezicima koji se interpretiraju imaju osobinu da se mnogo sporije izvršavaju od programa napisanih na jezicima koji se kompajliraju. Danas postoji veliki broj jezika koji kombinuju interpretiranje i kompajliranje (na primer programski jezik Java). Današnji programi koji se izvršavaju na računarima zahtevaju neko *okruženje* (eng. environment). Ova okruženje najčešće se odnose na *operativni sistem* (eng. operating system - OS) koji predstavlja program na našem računaru koji upravlja svim aktivnostima računara, upravlja hardverom, softverom i obezbeđuje potrebne servise našim programima. Na primer, ako naš program upisuje našto u fajl (datoteku), ta aktivnost zahteva poziv servisa operativnog sistema koji upravlja fajlovima, odnosno servisima koji čitaju i ispisuju sadržaj fajla na hard disk računara. Pored toga, operativni sistem pokreće program, upravlja memorijom koju program koristi, obezbeđuje pristup perifernim uređajima, kao što je na primer štampač. Poznati primeri operativnih sistema su Microsoft Windows i Linux za desktop računare i Android i iOS za mobilne uređaje.

U ovoj knjizi za ilustraciju osnovnih koncepata i algoritama programiranja korišćen je programski jezik C na operativnom sistemu Linux.

Jezik C je nastao između 1969. i 1973. godine kao okruženje za razvoj operativnog sistema UNIX, a kreirao ga je Dennis Ritchie [1]. Programi pisani u C-u se kompajliraju za odgovarajući operativni sistem, što znači da se program u mašinskom jeziku koji se dobija nakon kompajliranja programa napisanog u C-u može izvršavati samo na određenoj platformi (operativnom sistemu). Izvorni kôd C-programa piše se u tekstualnom fajlu koji ima ekstenziju c, kada se ovakvi fajlovi kompajliraju dobijaju se fajlovi koji predstavljaju izvršne verzije programa i njihova ekstenzija zavisi od operativnog sistema za koji se kompajliraju, može biti exe na Windows operativnom sistemu ili bez ekstenzije na Linux operativnom sistemu. Postoje i fajlovi sa programskom kôdom u progarmskom jeziku C koji imaju ekstenziju h to su fajlovi koji sadrže zaglavlja funkcija (eng. header files) što će biti objašnjeno u poglavlju 10.

Veći deo primera programa u ovoj knjizi koristi *standardni ulaz* i *standardni izlaz* koji predstavljaju deo operativnog sistema i preko njih je moguće

vršiti komunikaciju sa programom ili operativnim sistemom bez grafičkog korisničkog interfejsa (prozor sa ikonama i drugim grafičkim elementima koji se otvara na ekranu). Preko standardnog ulaza se vrši unos podataka u program kucanjem na tastaturi, dok se preko standardnog izlaza vrši prikaz nekog teksta iz programa. Ovakve programe često nazivamo programima za *komandnu liniju* ili *konzolu*. Pored toga što se uvek koriste na početku učenja programiranja, programi za komandnu liniju imaju vrlo veliku praktičnu primenu naročito u oblastima računarstva gde su vrlo bitne performanse izvršavanja programa koje su posebno narušene iscrtavanjem grafičkog korisničkog interfejsa.

1.2 Sintaksa programskog jezika

Da bismo naučili da pišemo programe u nekom programskom jeziku prvo moramo savladati (naučiti) njegovu sintaksu.

Sintaksu programskog jezika čine predefinisani izrazi u koje spadaju reči i znakovi interpunkcije i pravila za pisanje programa korišćenjem tih izraza. Primer jednog programa u programskom jeziku C prikazan je na listingu 1.1. Ovaj program na standardni izlaz ispisuje rečenicu `"Zdravo!"`.

Listing 1.1: Primer programa u C-u

```
1 # include <stdio.h>
2 int main(){
3       printf("Zdravo!\n");
4       return 0;
5 }
```

Značenje pojedinačnih linija kôda u ovom programu je sledeće:

`# include <stdio.h>`

- uvoz potrebnih biblioteka iz okruženja (već gotovih konstrukcija koje možemo da koristimo u našim programima), ovde je u pitanju standardna biblioteka za ulaz i izlaz, odnosno učitavanje podataka sa standardnog ulaza i ispis na standardni izlaz

`main()`

– definicija funkcije pod nazivom „main“ koja nema ulaznih parametara, što je označeno praznim zagradama

`{`

- početak bloka naredbi u kom je implementirana funkcija `main`

`printf("Zdravo!\n");`

- ispis na standardni izlaz jednog stringa (niz karaktera pod navodnicima), oznaka \n predstavlja oznaku za novi red. Funkcija printf je uvezena iz standardne