Predgovor

Ova knjiga predstavlja udžbenik za programski jezik *C* za široki krug čitalaca. Knjigu mogu da koriste i početnici u programiranju, ali je vrlo korisna i za profesionalne programere kojima jezik *C* nije osnovni programski jezik u profesionalnoj delatnosti.

Programski jezik *C* izložen je u obimu koji može da zadovoljava i naprednije neprofesionalne programere. Od sadržaja prateće standardne biblioteke funkcija prikazani su samo važniji delovi.

Jedini način da se nauči neki programski jezik jeste da se pišu programi na njemu. Ova knjiga je u potpunosti podređena tom osnovnom načelu.

Uvodno poglavlje 1 sadrži i informacije korisne pre svega za početnike: komande za obradu programa na jeziku *C* na računarima u tekstualnom i grafičkom okruženju i način prikazivanja algoritama pomoću dijagrama toka i Nasi-Šnajdermanovih dijagrama.

U poglavlju 2 obrađeni su podaci koji su predmet obrade u programima. Objašnjen je pojam tipa podataka, načini definisanja i korišćenja skalarnih brojčanih podataka. Da bi što pre mogli da se pišu potpuni programi u ovom poglavlju obrađena je i ulazna i izlazna konverzija brojčanih podataka. Naime, programi koji ne učitavaju podatke za obradu i ne ispisuju dobijene rezultate nemaju nikakvu upotrebnu vrednost.

Poglavlje 3 posvećeno je operatorima i načinima sastavljanja izraza pomoću njih. Tu su obrađeni operatori jezika *C* koji čitaocu s prosečnim matematičkim obrazovanjem ne bi trebalo da budu strani. Prikazivanje nekih specifičnijih operatora odloženo je za kasnije. Kao proširenje skupa operatora dat je i pregled važnijih standardnih bibliotečkih funkcija koje postoje u jeziku *C*. Odabirane su funkcije koje se odnose na opštepoznate pojmove iz matematike i na obradu već objašnjenih tipova podataka.

U poglavlju 4 prikazane su naredbe koje predstavljaju jedinične obrade u programima. Pored prostih naredbi obrađene su upravljačke strukture jezika *C* koje omogućavaju sastavljanje složenih programa na efikasan i pregledan način: uslovno (selekcije) i višestruko izvršavanje (ciklusi) delova programa.

Iz matematike dobro poznati nizovni tipovi: vektori, matrice i višedimenzionalni nizovi obrađeni su u poglavlju 5.

Tu se završava prva logička celina u prikazivanju jezika *C*. Trebalo bi da svako ko je početnik u programiranju dobro utvrdi ovaj deo materije i sam napiše što više programa, pre nego što produži dalje. Naredna poglavlja su za naprednije programere.

U poglavlju 6 obrađeni su pokazivači koji u jeziku *C* predstavljaju vrlo efikasan mehanizam za obradu nizova. Oni su neophodni za dinamičko stvaranje podataka u memoriji i za njihovu obradu.

Znakovni podaci, koji služe za predstavljanje tekstova, obrađeni su u poglavlju 7. Pored načina predstavljanja pojedinačnih znakova i niski znakova, prikazane su i važnije standardne bibliotečke funkcije za rad sa znakovima kao i za čitanje i pisanje tekstova.

Poglavlje 8 posvećeno je pisanju funkcija (potprograma) i opštoj strukturi programa u jeziku *C*. Potprogrami predstavljaju alat za realizaciju složenih obrada razbijanjem na skup jednostavnijih i razumljivijih radnji.

Pretprocesor jezika *C*, koji obavlja uvodnu obradu izvornog teksta programa pre samog prevođenja, objašnjen je u poglavlju 9.

U poglavlju 10 prikazani su strukturni tipovi podataka koji omogućavaju predstavljanje složenih apstraktnih podataka kao što su kompleksni brojevi, geometrijski likovi i slično. Strukturni tipovi su neophodni i pri obrazovanju dinamičkih struktura podataka kao što su, na primer, liste i stabla.

Na kraju, poglavlje 11 posvećeno je radu s datotekama. Datoteke služe za trajno uskladištavanje podataka na magnetnim diskovima.

Jezik *C* je dosta složen. Nisu svi detalji neophodni svakome, a naročito ne početnicima. Odeljci, ponegde cela poglavlja, u knjizi koji mogu da se preskoče u prvom čitanju, bilo zbog složenosti, bilo zbog manjeg značaja, obeleženi su sa △. Odeljci se, uglavnom, mogu preskakati nezavisno. Ako se preskoči jedan odeljak, to ne znači da mora da se preskoči i neki kasniji, obeleženi odeljak.

Rukovodeći se ranije istaknutim načelom o učenju jezika, ova knjiga, pored manjih primera u toku izlaganja, na kraju svakog poglavlja sadrži izvestan broj rešenih zadataka s detaljnim objašnjenjima. Kroz te zadatke skreće se pažnja na neke specifičnosti jezika *C* i daju ideje za elegantno i efikasno rešavanje nekih problema.

Pored rešenih zadataka ponuđen je i izvestan broj zadataka za samostalno rešavanje. Oni mogu da posluže kao polazne ideje za formulisanje složenijih problema.

Zainteresovani čitaoci mogu naći znatno veći broj zadataka u autorovoj zbirci *Rešeni zadaci iz programskog jezika C* ([7]).

Ova knjiga je više nego udžbenik za programski jezik *C*. Kroz rešene zadatke prikazane su i najvažnije tehnike koje se uopšteno koriste u računarskoj tehnici: binarno pretraživanje, uređivanje, obrada listi, manipulacije s tekstovima, itd.

U knjizi se maksimalno koriste principi strukturiranog programiranja. Pored toga, posebna pažnja je posvećena i inženjerskim aspektima programiranja. Nije dovoljno da program rešava zadati problem već treba da bude i „lep”, razumljiv, da troši što manje memorije i da bude što efikasniji.

Svi programi koji se nalaze u ovoj knjizi potpuni su u smislu da mogu da se izvršavaju na računaru. Provereni su na nekoliko računara korišćenjem nekoliko različitih prevodilaca za jezik *C*.

Izvorni tekstovi svih programa iz ove knjige mogu da se preuzmu sa Interneta, sa adrese `home.etf.rs/~kraus/knjige/`. Na istoj adresi objaviće se ispravke eventualnih, naknadno otkrivenih grešaka u knjizi. Svoja zapažanja čitaoci mogu da upute autoru elektronskom poštom na adresu `kraus@etf.rs`.

Laslo Kraus

Beograd, *januar 2020.*

Sadržaj

Predgovor ..... **v**

Sadržaj ..... **vii**

1 Uvod **1**

1.1 O programskom jeziku *C* 1
1.2 Obrada programa na jeziku *C* 2
1.2.1 Obrada programa u tekstualnom režimu 2
1.2.2 Obrada programa u grafičkom režimu 4
1.3 Prikazivanje algoritama 5
1.3.1 Dijagrami toka 5
1.3.2 Nasi-Šnajdermanovi dijagrami 7

2 Podaci **11**

2.1 Elementi jezika *C* 11
2.2 Komentari 12
2.3 Identifikatori 12
2.4 Tipovi podataka 13
2.5 Brojčani tipovi 14
2.5.1 Celobrojni tipovi podataka 14
2.5.2 Realni tipovi podataka 16
2.6 Brojčane konstante 16
2.6.1 Celobrojne konstante 16
2.6.2 Realne konstante 17
2.7 Logički podaci 18
2.8 Imenovane konstante 18
2.9 Nadimci tipova 19
2.10 Definisanje podataka 20
2.10.1 Definisanje promenljivih podataka 20
2.10.2 Definisanje nepostojanih podataka 20
2.10.3 Definisanje nepromenljivih podataka 21
2.11 Nabrojane konstante 21

2.12 Čitanje i pisanje podataka 23
2.12.1 Ulazna konverzija 23
2.12.2 Izlazna konverzija 26
2.13 Primeri potpunih programa 29

3 Operatori 37
3.1 Aritmetički operatori 38
3.1.1 Binarni operatori +, −, *, / i % 38
3.1.2 Unarni operatori + i − 39
3.1.3 Unarni operatori ++ i −− 39
3.2 Operatori za dodelu vrednosti 40
3.2.1 Inicijalizacija i dodela vrednosti 41
3.3 Pisanje složenih izraza 42
3.4 Zavisnost od redosleda izračunavanja operanada 42
3.5 Standardne funkcije 43
3.6 Matematičke funkcije 44
3.7 Rad s kompleksnim brojevima △ 46
3.8 Konverzija tipa 47
3.9 Relacioni operatori 48
3.10 Logički operatori 49
3.11 Uslovni operator 50
3.12 Operatori nad bitovima 51
3.13 Lančanje izraza 52
3.14 Veličina podataka 53
3.15 Pregled operatora 54
3.16 Zadaci 54
3.16.1 Površina trougla u ravni 54
3.16.2 Pakovanje i raspakivanje vremena 56

4 Naredbe 61
4.1 Prosta naredba 62
4.2 Sekvenca ili blok: {} 62
4.3 Doseg identifikatora 64
4.3.1 Ugnežđivanje dosega 64
4.4 Selekcije 67
4.4.1 Osnovna selekcija: if-else 67
4.4.1.1 Ugnežđivanje osnovnih selekcija 69
4.4.2 Generalizovana selekcija: if-else-if-else 71
4.4.3 Selekcija pomoću skretnice: switch 74
4.5 Ciklusi 76
4.5.1 Osnovni ciklus s izlazom na vrhu: while 77
4.5.2 Generalizovani ciklus s izlazom na vrhu: for 78
4.5.3 Ciklus s izlazom na dnu: do 81
4.6 Trajnost podataka 82

4.7 Skokovi ... 85
4.7.1 Iskakanje iz upravljačke strukture: `break` ... 85
4.7.1.1 Selekcija alternativnih grana ... 86
4.7.1.2 Ciklus s izlazom u sredini ... 88
4.7.2 Skok na kraj ciklusa: `continue` ... 90
4.7.3 Skok s proizvoljnim odredištem: `goto` ... 91
4.8 Zadaci ... 92
4.8.1 Rešavanje kvadratne jednačine ... 92
4.8.2 Računanje statističkih pokazatelja ... 95
4.8.3 Ispisivanje brojeva u binarnom sistemu ... 97

5 Nizovi ... 101
5.1 Definisanje nizova ... 101
5.2 Pristupanje elementima nizova ... 102
5.3 Inicijalizacija nizova ... 103
5.4 Nadimci nizovnih tipova ... 106
5.5 Veličina nizova ... 106
5.6 Zadaci ... 107
5.6.1 Skalarni proizvod dva vektora ... 107
5.6.2 Uređivanje nizova brojeva ... 109
5.6.3 Zadaci za samostalno rešavanje ... 114

6 Pokazivači ... 115
6.1 Definisanje pokazivača ... 116
6.2 Nepromenljivi i nepostojani podaci i pokazivači ... 118
6.3 Prikazivanje vrednosti pokazivača ... 119
6.4 Nadimci pokazivačkih tipova ... 120
6.5 Adresna aritmetika ... 121
6.6 Pokazivači i nizovi ... 122
6.7 Smeštanje višedimenzionalnih nizova u memoriju ... 123
6.8 Pokazivači na nizove ... 124
6.9 Složeni literali △ ... 127
6.10 Ograničeni pokazivači △ ... 128
6.11 Standardne funkcije △ ... 129
6.12 Dinamička dodela memorije ... 131
6.13 Dinamički nizovi ... 133
6.14 Zadaci ... 138
6.14.1 Unija skupova ... 138
6.14.2 Transponovanje matrice ... 141
6.14.3 Stepenovanje simetrične matrice △ ... 143
6.14.4 Zadaci za samostalno rešavanje ... 146

7 Znakovi ... 147
7.1 Kodiranje znakova ... 147
7.2 Konstante ... 149
7.2.1 Znakovne konstante ... 149
7.2.2 Niska-literali ... 150
7.2.3 Imenovane konstante ... 152

7.3 Matrice znakova ... 152
7.4 Ulazno-izlazna konverzija ... 153
7.5 Standardne funkcije ... 155
7.5.1 Čitanje i pisanje znakova bez konverzije ... 155
7.5.2 Ispitivanje znakova ... 156
7.5.3 Rad s niskama ... 156
7.5.4 Konverzija u brojčane tipove podataka ... 158
7.6 Zadaci ... 159
7.6.1 Prikazivanje tablice *ASCII* kodova ... 159
7.6.2 Brojanje slova i cifara ... 160
7.6.3 Uređivanje imena ... 162
7.6.4 Uređivanje imena gradova ... 165
7.6.5 Zadaci za samostalno rešavanje ... 172

8 Funkcije ... 175
8.1 Definisanje funkcija ... 175
8.1.1 Vrednost funkcije ... 175
8.1.2 Parametri funkcije ... 176
8.1.3 Telo funkcije ... 177
8.2 Pozivanje funkcija ... 178
8.3 Bočni efekti funkcija ... 180
8.4 Funkcije i nizovi ... 181
8.5 Nepromenljivi i nepostojani parametri ... 185
8.6 Ograničeni parametri △ ... 186
8.7 Globalni identifikatori ... 187
8.8 Deklaracija i definicija ... 191
8.8.1 Deklarisanje globalnih podataka ... 191
8.8.2 Deklarisanje funkcija ... 192
8.8.3 Primer deklarisanja globalnih identifikatora ... 193
8.9 Povezivanje identifikatora ... 194
8.10 Umetnute funkcije △ ... 196
8.11 Funkcije koje se ne vraćaju △ ... 198
8.12 Glavna funkcija ... 198
8.13 Rekurzivne funkcije ... 200
8.14 Pokazivači na funkcije ... 201
8.15 Funkcije s promenljivim brojem argumenata △ ... 204
8.16 Zadaci ... 206
8.16.1 Binarno pretraživanje uređenog niza ... 206
8.16.2 Raspakivanje celobrojnog podatka bit po bit ... 209
8.16.3 Ispisivanje Paskalovog trougla ... 211
8.16.4 Fuzija dinamičkih nizova ... 212
8.16.5 Zadaci za samostalno rešavanje ... 217

9 Pretprocesor ... 219
9.1 Umetanje sadržaja datoteke ... 219
9.2 Zamena leksičkih simbola ... 221
9.2.1 Unapred definisani identifikatori ... 223

9.3 Uslovno prevođenje 223
9.4 Razne direktive 225
9.5 Zadaci 226
9.5.1 Obrada polinoma 226

10 Strukture 233
10.1 Definisanje struktura 233
10.2 Upotreba struktura 236
10.3 Inicijalizacija struktura 239
10.4 Deklarisanje struktura △ 242
10.5 Dinamičke strukture podataka 243
10.6 Ugnežđene i bezimene strukture △ 245
10.7 Fleksibilne strukture △ 247
10.8 Unije 248
10.9 Polja bitova 250
10.10 Zadaci 250
10.10.1 Obrada tačaka u ravni 250
10.10.2 Obrada kompleksnih brojeva 254
10.10.3 Obrada jednostruko spregnutih listi 258
10.10.4 Zadaci za samostalno rešavanje 263

11 Datoteke 265
11.1 Otvaranje i zatvaranje datoteka 267
11.1.1 Pokazivač datoteke 267
11.1.2 Otvaranje datoteke 267
11.1.3 Zatvaranje datoteke 268
11.2 Rad s tekstualnim datotekama 268
11.2.1 Prenos znakova 268
11.2.2 Prenos podataka s konverzijom 270
11.3 Rad s binarnim datotekama 271
11.4 Pozicioniranje unutar datoteke (direktan pristup) 272
11.5 Signalizacija grešaka 273
11.6 Zadaci 273
11.6.1 Prepisivanje tekstualnih datoteka 273
11.6.2 Obrada datoteke o štedišama 276

Literatura 283

Indeks 285

1 Uvod

1.1 O programskom jeziku *C*

Programski jezik *C* jeste jezik opšte namene srednjeg nivoa, koji omogućava dosta intiman kontakt s hardverom računara. Poseduje strukturirane tipove podataka i upravljačke strukture što je karakteristika viših programskih jezika. S druge strane podržava manipulacije bitovima, korišćenje procesorskih registara, pristup podacima pomoću adrese i operatore orijentisane ka hardveru računara. Ovo su karakteristike nižih programskih jezika kao što su simbolički mašinski jezici.

Sintaksa jezika omogućava koncizno izražavanje i pisanje strukturiranih programa. Spada u grupu programskih jezika koji razlikuju mala i velika slova kod ključnih reči jezika i imena elemenata programa koje uvodi programer.

Programski jezik *C* projektovao je Denis Riči (*Dennis Ritchie*) 1972. godine u Belovim laboratorijama (*Bell Laboratories*). Osnovni cilj bio je sastavljanje jezika nezavisnog od računara, s karakteristikama viših programskih jezika, koji će moći da zameni simboličke mašinske (*assembler*) jezike koji su, i te kako, zavisni od računara.

Dugi niz godina osnovna definicija jezika *C* bio je Referentni priručnik (*Reference Manual*) u sastavu prvog izdanja knjige *The C Programming Language* čiji su autori Brajan V. Kernigan (*Brian W. Kernighan*) i Denis M. Riči ([3]). Varijanta jezika *C* koja je opisana u toj knjizi danas se naziva *K&R C*.

Prvi zvaničan standard za jezik *C*, takozvani *ANSI C*, izdao je Američki nacionalni institut za standarde (*American National Standards Institute*) 1989. godine pod brojem X3.159-1989. Ta varijanta jezika poznata je i pod imenom *C89*.

Od 1990. godine Međunarodna organizacija za standarde (*ISO – International Standards Organization*) prihvatila je *ANSI* standard kao svoj pod brojem ISO:IEC 9899:1990. Poznat je i pod imenom *ISO C*, ali koristi se i ime *C90* kao sinonim za *C89*.

Jezik *C89* zasnovan je na jeziku *K&R C* i nekim novim elementima koji su se već dosta raširenо koristili u praksi. Nije se posebno brinulo o kompatibilnosti zatečenih programa sa propisima standarda *C89*. Smatralo se da je u pitanju definisanje novog jezika.

Drugo izdanje *ISO* standarda, *C99*, objavljeno je 1999. godine. Ono je unelo znatan broj novih elemenata u jezik: komentare započete sa `//`, proizvoljan redosled deklarativnih i

izvršnih naredbi, umetnute funkcije, nizove s promenljivim dužinama itd. Neki od novih elemenata preuzeta su iz tog momenta važećeg standarda za jezik *C++*.

Neki od novih elemenata jezika koje je uveo treće izdanje *ISO* standarda, *C11* iz 2011. godine su: podrška za rad s kompleksnim brojevima, funkcije bez povratka, bezimene strukture i unije, podrška za višenitno izvršavanje programa itd.

Važeći standard u vreme pisanja ove knjige je treće izdanje *ISO* standarda, *C18* iz 2018. godine. Nije uveo nove jezičke elemente, već samo su otklonjene uočene neispravnosti unete su neke novine u pratećoj standardnoj biblioteci funkcija.

U nastavku ove knjige trenutno važeći standard za jezik *C* naziva se skraćeno *Standard*.

Jezik *C* tesno je povezan s operativnim sistemom *UNIX*. To je jezik koji obavezno postoji na bilo kom računaru pod operativnim sistemom *UNIX*. Sam operativni sistem *UNIX* većim delom pisan je na jeziku *C*, uključujući i prevodioca za jezik *C*.

Operativni sistem *UNIX* prvi je *de facto* standardni operativni sistem koji, pod raznim nazivima, nudi većina današnjih proizvođača računara. Jedna od najpoznatijih vaijanti je *LINUX* na ličnim računarima.

Pod operativnim sistemom *UNIX* kao standardni programski jezik za sistemsko programiranje, umesto nestandardnih simboličkih mašinskih jezika, nudi se programski jezik *C*.

Pošto je *C* kao programski jezik opšte namene vremenom stekao veliku popularnost, danas se nudi praktično pod svim operativnim sistemima. Štaviše, za većinu računara postoji veći broj prevodilaca od više različitih proizvođača. Danas vrlo popularni programski jezici *C++*, *C#* i *Java* korene sintakse i sementike vuku iz jezika C.

Programi pisani na jeziku *C* u velikoj su meri prenosivi s jednog računara na drugi. Većina programa može bez ikakvih izmena da se koristi na svim računarima koji podržavaju jezik *C*.

1.2 Obrada programa na jeziku *C*

Obrada programa sastoji se od sledeća četiri koraka:

- unošenje izvornog teksta programa u datoteku na disku,
- prevođenje izvornog teksta programa,
- povezivanje prevedenog oblika programa s potrebnim korisničkim i sistemskim potprogramima u izvodljivi oblik, i
- izvršavanje programa.

Obrada programa može da se izvodi u dva režima rada:

- tekstualnom režimu kada se koraci obrade izvode unošenjem komandi u obliku teksta, i
- grafičkom režimu korišćenjem prozora za unos teksta programa i izdavanje komandi.

U narednim odeljcima ukratko su prikazani osnovni elementi obrade programa u ova dva režima rada.

1.2.1 Obrada programa u tekstualnom režimu

Tekstualni režim obrade programa starijeg je datuma i danas se relativno retko koristi. Rad se odvija unošenjem komandi operativnom sistemu u obliku teksta preko tastature. Uneti

tekst, kao i poruke operativnog sistema ispisuju se na ekranu red po red. Izvršavanje programa, takođe, odvija se kroz dijalog preko tastature i ekrana.

Često korišćeni alat za obradu *C* programa je prevodilac *Microsoft C/C++ Compiler* koji se pod operativnim sistemom *Microsoft Windows* izvršava u komandnom prozoru (*Command Promt Window*). Komandni prozor treba da se otvori iz Start menija biranjem:

Start | Microsoft Visual Studio 20xx | Visual Studio Tools | Developer Command Prompt for VS20xx *ili…*
Start | Visual Studio 20xx | Developer Command Prompt for VS20xx

gde je 20xx broj verzije korišćenog *Visual Studija* (na primer, 2019) Na ovaj način će se pre otvaranja komandnog prozora podesiti parametri potrebni za pokretanje prevodioca *Microsoft C/C++ Compiler* i za izvršavanje korisničkih programa.

U slučaju rada na jeziku *C* ime datoteke izvornog teksta mora da ima proširenje **.c**. Za unošenje teksta može da se koristi bilo koji urednik teksta koji stoji na raspolaganju. Na primer, komanda za unošenje i obradu izvornog teksta programa u datoteci *imeprog*.**cpp** može biti:

```
notepad imeprog.c
```

Po završetku sastavljanja izvornog teksta programa, komandom:

```
cl imeprog.c
```

prevodi se izvorni tekst programa na jeziku *C* iz datoteke *imeprog*.**c**. Rezultat prevođenja smešta se u datoteku *imeprog*.**obj**, i odmah se stvara i povezani oblik programa koji se smešta u datoteku *imeprog*.**exe**.

U slučaju sastavljanja velikog programskog sistema vrlo često se izvorni tekst glavnog programa (glavne funkcije) i svih potrebnih potprograma (funkcija) smešta u više datoteka. U tom slučaju potrebno je da se preskoči povezivanje prilikom prevođenja sadržaja datoteka u kojima se nalaze samo funkcije. To se postiže dodavanjem opcije **/c** u komandi **cl**:

```
cl /c imeprog.c
```

U komandi **cl** može da bude naveden i čitav niz imena datoteka od kojih neke mogu da sadrže izvorne tekstove, a neke prevedene oblike programskih modula. Pomoću komande:

```
cl ime1.c ime2.obj … imeN.
```

prevodiće se sadržaji datoteka čija se imena završavaju na **.c** i rezultati prevođenja smeštaće se u odgovarajuće datoteke sa sufiksom imena **.obj**. Posle toga, povezuju se programski moduli iz svih novostvorenih datoteka kao i iz datoteka čija se imena u prethodnoj komandi završavaju sa **.obj**. Povezani oblik celokupnog programskog paketa smešta se u datoteku imena *ime1*.**exe**.

Na kraju, izvršavanje programa postiže se komandom koja se sastoji samo od imena datoteke koja sadrži povezani oblik programa:

```
imeprog
```

Posle jedne od ove dve komande počinje izvršavanje programa. U toku rada programa korisnik treba da preko tastature unosi podatke po redosledu kako ih očekuje program sa „standardnog ulaza" računara. Na ekranu će se pojaviti podaci koji se unose preko tastature kao i podaci koje program šalje na „standardni izlaz" računara.

Pod operativnim sistemom *Windows* postoji vrlo elegantan način za „skretanje" standardnog ulaza i/ili standardnog izlaza. Time se postiže da se podaci čitaju iz neke datoteke umesto s tastature, odnosno da se upisuju u neku datoteku umesto da se ispisuju na ekranu. To se postiže izvršavanjem programa pomoću komande oblika:

```
imeprog <podaci >rezult
```