Predgovor

Ova knjiga predstavlja udžbenik za programski jezik *C#* za široki krug čitalaca. Knjigu mogu da koriste i početnici u programiranju, ali poznavanje osnovnih pojmova iz objektno-orijentisanog programiranja i programskih jezika *C/C++* ili *Java* znatno olakšava da se savlada materija iz ove knjige.

Programski jezik *C#* izložen je u obimu koji može da zadovoljava i naprednije neprofesionalne programere. Od ogromne standardne biblioteke klasa, koja prati jezik *C#*, objašnjeni su samo delovi koji su potrebni za efikasno programiranje pri rešavanju većine problema relativno visoke složenosti.

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 informacije korisne pre svega za početnike: osnovne pojmove objektno-orijentisanog programiranja, glavne osobine programskog jezika *C#* i načine obrade programa u tekstualnom i grafičkom okruženju.

U poglavlju 2 obrađeni su podaci koji su predmet obrade u programima. Objašnjen je pojam tipa podataka, kao i načini definisanja i korišćenja skalarnih brojčanih, znakovnih, logičkih i tekstualnih podataka. Da bi što pre mogli da se pišu potpuni programi u ovom poglavlju su obrađene i ulazna i izlazna konverzija brojčanih podataka. Naime, programi koji ne čitaju početne podatke i ispisuju 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 je odloženo za kasnije. Kao proširenje skupa operatora dat je i pregled važnijih standardnih bibliotečkih funkcija koje postoje u jeziku *C#*. Odabrane 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 i složene naredbe jezika *C#* koje omogućavaju sastavljanje složenih programa na efikasan i pregledan način: uslovno izvršavanje (selekcije), višestruko izvršavanje (ciklusi) delova programa i upravljačke naredbe (skokovi).

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

Izučavanje objektno-orijentisanog programiranja počinje u poglavlju 6 o klasama. Klase su složeni tipovi podataka koji programerima omogućuju modeliranje objekata iz realnog života. Sastoje se od polja čije vrednosti čine stanja primeraka klasa i metoda koje definišu moguće načine korišćenja tih polja. Primerci klasa su podaci klasnih tipova i nazivaju se

objektima. Jedan od važnih elemenata ispravnog rada programa je pravilna inicijalizacija objekata.

Programski jezik *C#* omogućava grupisanje srodnih klasa u prostore imena. U poglavlju 7 obrađeni su prostori imena: njihovo formiranje i korišćenje. Prikazani su i važniji elementi dva najčešće korićena standardna prostora imena.

Poglavlje 8 posvećeno je sledećem fundamentalnom konceptu objektno-orijentisanog programiranja, nasleđivanju. U grupama podataka koje su predstavljene klasama mogu da se uoče podgrupe sa nekim specifičnijim osobinama. Te podgrupe, predstavljene potklasama, nasleđuju osobine širih grupa, natklasa. Član podgrupe uvek može da se koristi i kao član opštije nadgrupe. U ovom poglavlju obrađeni su i interfejsi, specijalni apstraktni tipovi koji samo definišu određene osobine koje klase koje ih ostvaruju treba da poseduju, a ne i način kako se te osobine ostvaruju.

U poglavlju 9 obrađeni su vrednosni tipovi, tipovi čije promenljive podatke sadrže neposredno. Za razliku od njih, promenljive nizovnih i klasanih tipova tipova sadrže samo upućivače na mesta u memoriji gde se nalaze elementi nizova, odnosno primerci klasa.

Jezik *C#* omogućava da rukovanje izuzecima (greškama) vremenski ne opterećuje program sve dok se izuzetak ne pojavi. Kada se izuzetak pojavi, kontrola se automatski predaje jednom od rukovalaca izuzecima koje je definisao programer. Način rukovanja izuzecima prikazan je u poglavlju 10.

U poglavlju 11 objašnjeni su generički tipovi i metode. Generičke klase i metode omogućavaju pisanje klasa i metoda za obradu podataka, čiji tipovi nisu poznati u vreme njihovog pisanja, na bezbedan način sa stanovišta provera ispravnog korišćenja tipova podataka. Generički interfejsi, na sličan način, opisuju očekivane osobine budućih klasa u uslovima kada tipovi pojedinih korišćenih podataka nisu poznati. Na kraju ovog poglavlja su ukratko objašnjene i dve najčešće korišćene standardne generičke klase za rad sa zbirkama podataka.

U poglavlju 12 izloženi su osnovni koncepti konkurentne obrade, obrade koja se sastoji od istovremenog odvijanja više tokova kontrole, više istovremenih obrada. Ti tokovi kontrola nazivaju se niti, čija je korektna sinhronizacija osnova dobrog programa s konkurentnom obradom.

Rukovodeći se već istaknutim načelom o učenju jezika, ova knjiga, pored manjih primera u toku izlaganja, na kraju svakog poglavlja sadrži nekoliko rešenih zadataka sa detaljnim objašnjenjima. Više zadataka može da se nađe u autorovoj zbirci *Rešeni zadaci iz programskog jezika C#* (videti [8]). Kroz zadatke u ovoj knjizi i u zbirci skreće se pažnja na neke specifičnosti jezika *C#* i daju se ideje za elegantno i efikasno rešavanje nekih problema.

Ova knjiga je više nego udžbenik za programski jezik *C#.* Kroz rešene zadatke prikazane su i primene osnovnih elemenata objektno-orijentisanog programiranja, a na primerima koji se uopšteno sreću u računarskoj tehnici: obradi redova, stekova, listi, tekstova itd. Posebna pažnja posvećena je i inženjerskim aspektima programiranja. Nije dovoljno da program rešava zadati problem, već treba da bude i „lep", razumljiv, da se lako održava, 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, oktobar 2016.

Sadržaj

Predgovor ..... **v**

Sadržaj ..... **vii**

1 Uvod **1**

1.1 O programskom jeziku *C#* 1
1.2 Uvod u objektno-orijentisano programiranje 2
1.2.1 Apstrakcija 3
1.2.2 Učaurivanje 4
1.2.3 Nasleđivanje 5
1.2.4 Polimorfizam 5
1.2.5 Ponovno korišćenje koda 6
1.3 Osobine jezika *C#* 6
1.4 Obrada programa na jeziku *C#* 7
1.4.1 Obrada programa u tekstualnom režimu 7
1.4.2 Obrada programa u grafičkom režimu 8

2 Podaci **11**

2.1 Elementi jezika *C#* 11
2.2 Identifikatori 12
2.3 Tipovi podataka 13
2.4 Brojčani tipovi 15
2.4.1 Celobrojni tipovi podataka 15
2.4.2 Realni tipovi podataka 16
2.5 Znakovni podaci 17
2.6 Logički podaci 18
2.7 Niske 18
2.8 Definisanje podataka 20
2.9 Čitanje i pisanje podataka 21
2.9.1 Čitanje podataka 22
2.9.2 Pisanje podataka 23
2.10 Primeri potpunih programa 26

3 Operatori 33
3.1 Aritmetički operatori 34
3.2 Operatori za dodelu vrednosti 36
3.2.1 Inicijalizacija i dodela vrednosti 37
3.3 Pisanje složenih izraza 37
3.4 Matematičke funkcije 38
3.5 Konverzija tipa 39
3.6 Relacioni operatori 41
3.7 Logički operatori 41
3.8 Uslovni operator 43
3.9 Operatori po bitovima 43
3.10 Operatori za niske 45
3.11 Konstantni izrazi 46
3.12 Podrazumevane vrednosti tipova 46
3.13 Dohvatanje imena elemenata programa 46
3.14 Provera prekoračenja △ 47
3.15 Pregled operatora 47
3.16 Zadaci 48
3.16.1 Površina trougla u ravni 48
3.16.2 Pakovanje i raspakivanje vremena 50

4 Naredbe 55
4.1 Prosta naredba 55
4.2 Sekvenca ili blok: `{}` 56
4.2.1 Provera prekoračenja △ 56
4.3 Doseg identifikatora 57
4.4 Selekcije 59
4.4.1 Osnovna selekcija: `if-else` 59
4.4.2 Generalizovana selekcija: `if-else-if-else` 61
4.4.3 Selekcija pomoću skretnice: `switch` 62
4.5 Ciklusi 63
4.5.1 Osnovni ciklus s izlazom na vrhu: `while` 64
4.5.2 Generalizovani ciklus s izlazom na vrhu: `for` 65
4.5.3 Ciklus s izlazom na dnu: `do` 66
4.6 Skokovi 67
4.6.1 Iskakanje iz upravljačke strukture: `break` 67
4.6.1.1 Ciklus s izlazom u sredini 68
4.6.2 Skok na kraj ciklusa: `continue` 69
4.6.3 Skok s proizvoljnim odredištem: `goto` 69
4.6.3.1 Prelazak iz odeljka u odeljak u selekciji sa skretnicom 70
4.7 Zadaci 71
4.7.1 Kvadratna jednačina 71
4.7.2 Statistički pokazatelji 73
4.7.3 Ispisivanje brojeva u binarnom sistemu 74

5 Nizovi ... 77
5.1 Definisanje nizova ... 77
5.2 Inicijalizacija nizova ... 79
5.3 Pristupanje elementima niza: `[]` ... 80
5.4 Dohvatanje dužine niza: `Length`, `Rank`, `GetLength` ... 80
5.5 Dodela vrednosti nizovnih promenljivih: = ... 81
5.6 Upoređivanje nizovnih promenljivih: ==, `!=` ... 82
5.7 Razuđeni nizovi ... 82
5.8 Obilazak elemenata niza: `foreach` ... 86
5.9 Sakupljač otpadaka ... 87
5.10 Zadaci ... 88
5.10.1 Skalarni proizvod dva vektora ... 88
5.10.2 Uređivanje nizova brojeva ... 90
5.10.3 Unija skupova ... 94
5.10.4 Transponovanje matrice ... 97
5.10.5 Stepenovanje simetrične matrice ... 99

6 Klase ... 103
6.1 Definisanje klasa ... 104
6.2 Pristupanje članovima klasa ... 105
6.3 Polja klasa ... 105
6.4 Metode klasa ... 107
6.4.1 Definisanje metoda ... 107
6.4.1.1 Povratna vrednost metode ... 107
6.4.1.2 Parametri metode ... 107
6.4.1.3 Skriveni parametar metode ... 108
6.4.1.4 Telo metode ... 108
6.4.2 Pozivanje metode ... 110
6.4.3 Prenošenje argumenata u metode ... 112
6.4.4 Rekurzivne metode ... 115
6.4.5 Podrazumevani argumenti metoda ... 116
6.4.6 Metode s promenljivim brojem argumenata △ ... 117
6.4.7 Preopterećivanje imena metoda ... 118
6.5 Konstruktori ... 120
6.6 Objekti ... 122
6.6.1 Definisanje objekata ... 122
6.6.2 Tok stvaranja objekata ... 124
6.6.3 Uništavanje objekata ... 124
6.6.4 Tekstualni opis objekata ... 124
6.7 Statički članovi klasa ... 126
6.7.1 Definisanje statičkih članova ... 126
6.7.2 Inicijalizacija klasa ... 128
6.8 Glavna metoda ... 129
6.9 Dijagrami klasa ... 130
6.10 Svojstva ... 133
6.11 Indekseri ... 136

6.12 Statičke klase ... 140
6.12.1 Proširujuće metode △ ... 140
6.13 Ugnežđene klase ... 141
6.14 Dinamičke strukture podataka ... 143
6.15 Preopterećivanje operatora ... 145
6.15.1 Operatorske metode ... 146
6.15.2 Preopterećivanje operatora s bočnim efektima ... 148
6.15.3 Preopterećivanje operatora (*tip*) ... 149
6.15.3.1 Implicitna konverzija tipa ... 150
6.15.4 Operatori `true` i `false` △ ... 152
6.15.5 Preopretećivanje operatora `&&` i `||` △ ... 153
6.16 Zaobilaženje `null` upućivača △ ... 154
6.17 Zadaci ... 156
6.17.1 Tačke u ravni ... 156
6.17.2 Nizovi tačaka u ravni ... 159
6.17.3 Materijalne tačke u prostoru ... 163
6.17.4 Police s teglama ... 168
6.17.5 Redovi tegli neograničenog kapaciteta ... 174
6.17.6 Vremenski intervali ... 179
6.17.7 Polinomi s realnim koeficijentima ... 182

7 Prostori imena ... 189

7.1 Definisanje prostora imena ... 189
7.2 Korišćenje članova prostora imena ... 190
7.3 Ugnežđivanje prostora imena △ ... 191
7.4 Nadimci ... 193
7.5 Uvoženje statičkih članova klasa ... 194
7.6 Dijagrami klasa ... 195
7.7 Prostor imena `System` ... 196
7.7.1 Uslužna klasa za matematičke funkcije: `Math` ... 196
7.7.2 Klasa za nepromenljive niske: `String` ... 196
7.7.3 Klasa za generatore pseudoslučajnih brojeva: `Random` △ ... 199
7.7.4 Klasa za konzolu: `Console` △ ... 200
7.7.5 Klasa za okruženje programa: `Environment` △ ... 201
7.7.6 Klasa za sakupljač otpadaka: `GC` △ ... 201
7.8 Prostor imena `System.Text` ... 201
7.8.1 Klasa za promenljive niske: `StringBuilder` ... 202
7.9 Zadaci ... 204
7.9.1 Ravan s obojenim krugovima ... 204

8 Nasleđivanje ... 213

8.1 Izvedene klase ... 214
8.1.1 Definisanje izvedenih klasa ... 214
8.1.2 Korišćenje članova izvedenih klasa ... 217
8.1.3 Sakrivanje članova osnovne klase ... 218
8.1.4 Korišćenje članova izvan izvedene klase ... 221
8.1.5 Stvaranje objekata izvedenih klasa ... 222

8.2 Kompatibilnost tipova ... 224
8.3 Nizovi ... 227
8.4 Polimorfizam ... 228
8.5 Klasa `Object` ... 231
8.6 Apstraktne klase ... 233
8.7 Interfejsi ... 234
8.7.1 Definisanje interfejsa ... 235
8.7.2 Ostvarivanje interfejsa ... 235
8.7.2.1 Eksplicitno ostvarivanje članova interfejsa △ ... 237
8.7.3 Višestruko nasleđivanje interfejsa ... 238
8.7.4 Izvedeni interfejsi ... 238
8.7.5 Apstraktna klasa protiv interfejsa ... 239
8.8 Primer složenijeg dijagrama klasa ... 240
8.9 Kloniranje objekata ... 241
8.10 Delegati ... 244
8.10.1 Definisanje, stvaranje i pozivanje delegata ... 245
8.10.2 Bezimene metode i lambda izrazi ... 248
8.11 Događaji ... 251
8.12 Zadaci ... 253
8.12.1 Geometrijske figure u ravni ... 253
8.12.2 Uređivanje uporedivih objekata ... 260
8.12.3 Predmeti koji mogu da se kloniraju ... 267
8.12.4 Nizovi celih brojeva ... 275
8.12.5 Redovi objekata neograničenog kapaciteta ... 277

9 Vrednosni tipovi ... 281

9.1 Strukture ... 281
9.1.1 Definisanje struktura ... 281
9.1.2 Strukture za proste tipove ... 284
9.1.2.1 Brojčani tipovi ... 284
9.1.2.2 Znakovni tip ... 285
9.1.2.3 Logički tip ... 285
9.2 Nabrajanja ... 286
9.2.1 Klasa `Enum` ... 287
9.3 Zadaci ... 288
9.3.1 Špilovi karata za igru ... 288

10 Izuzeci ... 293

10.1 Klase za izuzetke ... 294
10.2 Prijavljivanje izuzetaka ... 295
10.3 Rukovanje izuzecima ... 296
10.3.1 Naredba `try` ... 296
10.3.2 Izvršavanje naredbe `try` ... 297
10.3.3 Filtri za izuzetke ... 299
10.4 Zadaci ... 300
10.4.1 Vektori zadatog opsega indeksa ... 300
10.4.2 Izračunavanje određenog integrala ... 304

11 Generički tipovi i metode 315
11.1 Tipovne promenljive 315
11.2 Generički tipovii 316
11.2.1 Definisanje generičkih klasa i interfejsa 316
11.2.2 Konkretizacija generičkih tipova 317
11.2.3 Uskladištavanje podataka u generičke objekte 318
11.2.4 Generičke izvedene klase i ugnežđene klase 320
11.3 Generičke metode 320
11.3.1 Definisanje generičkih metoda 320
11.3.2 Pozivanje generičkih metoda 321
11.3.3 Zaključivanje tipovnih argumenata 322
11.3.4 Generičke metode u generičkim klasama 324
11.4 Ograničavanje tipovnih parametara 325
11.5 Zbirke podataka 328
11.5.1 Prostor imena `System.Collections` 328
11.5.1.1 Obilazak negeneričkih zbirki `foreach` ciklusom 328
11.5.2 Prostor imena `System.Collections.Generic` 331
11.5.2.1 Obilazak generičkih zbirki `foreach` ciklusom 331
11.5.3 Standardni tipovi za generičke zbirke 333
11.6 Tipovi koji prihvataju `null` Δ 336
11.7 Zadaci 338
11.7.1 Generički redovi neograničenog kapaciteta 338
11.7.2 Poboljšana klasa za generičke redove objekata 341
11.7.3 Generički nizovi geometrijskih figura u ravni 343

12 Niti 351
12.1 Niti u jeziku *C#* 352
12.2 Stvaranje i izvršavanje niti – klasa `Thread` 353
12.2.1 Stvaranje objekta niti 353
12.2.2 Svojstva niti 354
12.2.3 Pokretanje niti 356
12.2.4 Dohvatanje trenutne niti 357
12.2.5 Zahtev za deblokiranje niti 357
12.2.6 Zaustavljanje niti na određeno vreme 358
12.2.7 Ustupanje procesora 359
12.2.8 Čekanje da se druga nit završi 359
12.3 Sinhronizacija niti – klasa `Monitor` 361
12.3.1 Kritična sekcija 361
12.3.2 Modifikator `volatile` Δ 363
12.3.3 Ulazak u kritičnu sekciju Δ 364
12.3.4 Izlazak iz kritične sekcije Δ 365
12.3.5 Čekanje na signal 366
12.3.6 Slanje signala 367
12.4 Zadaci 369
12.4.1 Zbirka s konkurentnim izračunavanjem ukupne veličine 369
12.4.2 Problem proizvođači/potrošači 374

Literatura **385**

Indeks **387**

1 Uvod

1.1 O programskom jeziku *C#*

Programski jezik *C#* viši je programski jezik opšte namene koji u potpunosti podržava paradigmu objektno-orijentisanog programiranja.

C# je više od programskog jezika koji ima određenu sintaksu i semantiku. Ona je pravo programsko okruženje koje pomoću ogromne biblioteke standardnih programskih modula podržava obrade bez kojih danas ne može da se zamisli nijedan ozbiljan program. Tu spadaju komunikacija sa korisnikom preko prozora, višenitna obrada, rad preko *Interneta*, troslojna arhitektura rada s bazama podataka itd. Standardi programskih jezika do pojave jezika *C#* te elemente nisu obuhvatali. Svaki proizvođač je uvodio nestandardna rešenja za njih, ograničavajući time prenosivost programa s jedne na drugu platformu.

Programski jezik *C#* je srodnik jezika *C++* koji je direktni potomak jezika *C*, a u velikoj meri liči i na jezik *C#*.

Pogramski jezik *C* pojavio se početkom 1970-tih godina i u osnovama je promenio dotadašnji, nesistematizovani, stil programiranja. Kroz podršku za *strukturirano programiranje* obezbedio je da mogu da se napišu, razumeju i održavaju znatno veći programski sistemi nego pre. Pošto se u centru pažnje nalaze postupci koji se primenjuju na podatke, ta tehnika programiranja naziva se i *proceduralno programiranje*.

Pošto su jezik *C* projektovali programeri koji su i sami pisali programe, drugi programeri su ga rado prihvatili i uskoro je postao jedan od najrasprostranjenijih programskih jezika, koji se koristi i danas. Za autora jezika *C* smatra se Denis Riči (*Dennis Ritchie*) iz Belovih laboratorija (*Bell Laboratories*).

Početkom 1980-tih godina računarski programi narasli su do takvih veličina da su se tehnike proceduralnog programiranja pokazale neefikasnim. Rešenje se našlo u paradigmi *objektno-orijentisanog programiranja*, kod koje se u centru pažnje nalaze podaci (objekti) na koje se primenjuju neki postupci. Od programskih jezika koji su projektovani da podržavaju nove tehnike programiranja najveći uspeh imala je nadogradnja jezika *C*, koja je danas poznata pod imenom *C++* (++ je u jeziku *C* operator povećavanja!). Preko 95% jezika *C* usvojeno je bez izmena i u jeziku *C++*.

C++ je hibridni jezik u smislu što, pored novijeg objektno-orijentisanog programiranja, podžava i starije proceduralno programiranje.

Za autora jezika *C++* smatra se Bjarn Strostrup (*Bjarne Stroustrup*).