PROGRAMIRANJE 1

Milena Vujošević Janičić, Jovana Kovačević,
Danijela Simić, Anđelka Zečević,
Aleksandra Kocić

PROGRAMIRANJE 1
Zbirka zadataka sa rešenjima

Beograd
2019.

Autori:
dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu
dr Jovana Kovačević, docent na Matematičkom fakultetu u Beogradu
dr Danijela Simić, asistent na Matematičkom fakultetu u Beogradu
Anđelka Zečević, asistent na Matematičkom fakultetu u Beogradu
Aleksandra Kocić, asistent na Matematičkom fakultetu u Beogradu

PROGRAMIRANJE 1
Zbirka zadataka sa rešenjima

Izdavač: Matematički fakultet Univerziteta u Beogradu. Studentski trg 16, Beograd.
Za izdavača: *prof. dr Zoran Rakić*, dekan

Recenzenti:
dr Gordana Pavlović-Lažetić, redovni profesor na Matematičkom fakultetu u Beogradu
dr Dragan Urošević, naučni savetnik na Matematičkom institutu SANU

Obrada teksta, crteži i korice: *autori*.
Štampa: SKRIPTA Internacional, Beograd. Tiraž 100.

CIP Каталогизација у публикацији
Народна библиотека Србије, Београд

004.42(075.8)(076)
004.432.2C(075.8)(076)

PROGRAMIRANJE 1 : zbirka zadataka sa rešenjima / Milena Vujošević Janičić ... [et al.]. - 1. izd. - Beograd : Univerzitet u Beogradu, Matematički fakultet, 2019 (Beograd : Skripta Internacional). - VII, 474 str. : ilustr. ; 25 cm

Tiraž 100.

ISBN 978-86-7589-139-0

1. Вујошевић Јаничић, Милена 1980- [аутор]

a) Програмирање - Вежбе b) Програмски језик "C"- Задаци
COBISS.SR-ID 279933452

Sadržaj

Predgovor

U okviru kursa *Programiranje 1*, koji se drži na prvoj godini na svim smerovima na Matematičkom fakultetu, vežbaju se zadaci koji imaju za cilj da studentima pomognu da nauče osnovne algoritme i strukture podataka koji se sreću u imperativnim programskim jezicima. Ova zbirka predstavlja objedinjen skup zadataka sa vežbi i praktikuma ovog kursa, kao i primere zadataka sa održanih ispita. Sva rešenja su data u programskom jeziku C, ali se većina zadataka može koristiti za vežbanje proizvoljnog imperativnog programskog jezika. Elektronska verzija zbirke i propratna rešenja u elektronskom formatu, dostupna su besplatno u skladu sa navedenom licencom i mogu se naći na adresi `http://www.programiranje1.matf.bg.ac.rs/zbirka`.

Zbirka je podeljena u četiri poglavlja. U prvom pogavlju obrađene su uvodne teme koje obuhvataju osnovne elemente imperativnog programiranja koje se koriste u rešavanju svih ostalih zadataka u zbirci. Uvodne teme uključuju osnovne tipove podataka, elementranu komunikaciju sa korisnikom, građenje izraza, upotrebu naredbi dodele i naredbi koje regulišu kontrolu toka programa (sekvenca, selekcija i iteracija) uključujući i izdvajanje logičkih celina u funkcije. Drugo poglavlje je posvećeno radu sa naprednijim tipovima podataka: nizovima (uključujući niske i višedimenzione nizove), pokazivačima i strukturama. Treće poglavlje posvećeno je dodatnim tehnikama koje se koriste za komunikaciju sa korisnikom. Obrađen je rad sa argumentima komandne linije, kao i rad sa datotekama. Dodatak sadrži primere dva ispitna roka iz jedne akademske godine. Većina zadataka je rešena, a teži zadaci su obeleženi zvezdicom.

Autori velikog broja zadataka ove zbirke su ujedno i autori same zbirke, ali postoje i zadaci za koje se ne može tačno utvrditi ko je originalni autor jer su zadacima davali svoje doprinose različiti asistenti koji su držali vežbe iz ovog kursa tokom prethodnih godina. Zbog toga smatramo da je naš osnovni doprinos što smo objedinili, precizno formulisali, rešili i detaljno iskomentarisali sve najvažnije zadatke koji su potrebni za uspešno savlađivanje koncepata koji se obrađuju u okviru kursa. Takođe, formulacije zadataka smo obogatili primerima koji upotpunjuju razumevanje zahteva zadataka i koji omogućavaju čitaocu

zbirke da proveri sopstvena rešenja. Primeri su dati u obliku testova i interakcija sa programom. Testovi su svedene prirode i obuhvataju samo jednostavne ulaze i izlaze iz programa. Interakcija sa programom obuhvata naizmeničnu interakciju čovek-računar u kojoj su ulazi i izlazi isprepletani. U zadacima koji zahtevaju rad sa argumentima komandne linije, navedeni su i primeri poziva programa, a u zadacima koji demonstriraju rad sa datotekama, i primeri ulaznih ili izlaznih datoteka. Test primeri koji su navedeni uz ispitne zadatke u dodatku su oni koji su korišćeni u okviru testiranja i ocenjivanja studentskih radova na ispitima.

Neizmerno zahvaljujemo recenzentima, Gordani Pavlović Lažetić i Draganu Uroševiću, na veoma pažljivom čitanju rukopisa i na brojnim korisnim sugestijama koje su unapredile kvalitet zbirke. Takođe, zahvaljujemo studentima koji su svojim aktivnim učešćem u nastavi pomogli i doprineli uobličavanju ovog materijala kao i svim kolegama koje su držale vežbe na ovom kursu.

Svi komentari i sugestije na sadržaj zbirke su dobrodošli i osećajte se slobodnim da ih pošaljete elektronskom poštom bilo kom autoru [1].

Autori

[1] Adrese autora su: `milena`, `jovana`, `danijela`, `andjelkaz`, `aleksandra_kocic`, sa nastavkom `@matf.bg.ac.rs`

1

Osnovni elementi imperativnog programiranja

1.1 Izrazi

Zadatak 1.1.1 Napisati program koji na standardni izlaz ispisuje poruku `Zdravo svima!`.

Test 1

```
Izlaz:
 Zdravo svima!
```

Zadatak 1.1.2 Napisati program koji za uneti ceo broj ispisuje njegov kvadrat i njegov kub.

Primer 1

```
Interakcija sa programom:
 Unesite ceo broj: 4
 Kvadrat: 16
 Kub: 64
```

Primer 2

```
Interakcija sa programom:
 Unesite ceo broj: -14
 Kvadrat: 196
 Kub: -2744
```

Zadatak 1.1.3 Napisati program koji za uneta dva cela broja x i y ispisuje njihov zbir, razliku, proizvod, ceo deo pri deljenju prvog broja drugim brojem i ostatak pri deljenju prvog broja drugim brojem. Napomena: *Pretpostaviti da je unos ispravan.*[1]

[1] U zadacima sa ovom napomenom treba pretpostaviti da se na ulazu zadaje očekivani broj vrednosti očekivanog tipa u, gde je primereno, odgovarajućem formatu.

Primer 1

```
Interakcija sa programom:
 Unesite vrednost broja x: 7
 Unesite vrednost broja y: 2
 7 + 2 = 9
 7 - 2 = 5
 7 * 2 = 14
 7 / 2 = 3
 7 % 2 = 1
```

Primer 2

```
Interakcija sa programom:
 Unesite vrednost promenljive x: -3
 Unesite vrednost promenljive y: 8
 -3 + 8 = 5
 -3 - 8 = -11
 -3 * 8 = -24
 -3 / 8 = 0
 -3 % 8 = -3
```

Zadatak 1.1.4 Napisati program koji pomaže kasirki da izračuna ukupan račun ako su poznate cene dva kupljena artikla. Cene artikala su pozitivni celi brojevi. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite cenu prvog artikla: 173
 Unesite cenu drugog artikla: 2024
 Ukupna cena: 2197
```

Primer 2

```
Interakcija sa programom:
 Unesite cenu prvog artikla: 384
 Unesite cenu drugog artikla: 555
 Ukupna cena: 939
```

Zadatak 1.1.5 Napisati program koji za unetu količinu jabuka u kilogramima i unetu cenu po kilogramu ispisuje ukupnu cenu date količine jabuka. Obe ulazne vrednosti su pozitivni celi brojevi. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite kolicinu jabuka (u kg): 6
 Unesite cenu (u dinarima): 82
 Molimo platite 492 dinara.
```

Primer 1

```
Interakcija sa programom:
 Unesite kolicinu jabuka (u kg): 10
 Unesite cenu (u dinarima): 93
 Molimo platite 930 dinara.
```

Zadatak 1.1.6 Napisati program koji pomaže kasirki da obračuna kusur koji treba da vrati kupcu. Za unetu cenu artikla, količinu artikla i iznos koji je kupac dao, program treba da ispiše vrednost kusura. Sve ulazne vrednosti su pozitivni celi brojevi. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite cenu, kolicinu i iznos:
 132 2 500
 Kusur: 236 dinara
```

Primer 2

```
Interakcija sa programom:
 Unesite cenu, kolicinu i iznos:
 59 6 2000
 Kusur: 1646 dinara
```

Zadatak 1.1.7 Napisati program koji za uneta vremena poletanja i sletanja

aviona izražena u satima i minutima ispisuje dužinu trajanja leta. NAPOMENA: *Pretpostaviti da su poletanje i sletanje u istom danu kao i da su sve vrednosti ispravno unete.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vreme poletanja: 8 5
 Unesite vreme sletanja: 12 41
 Duzina trajanja leta: 4 h i 36 min
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vreme poletanja: 13 20
 Unesite vreme sletanja: 18 45
 Duzina trajanja leta: 5 h i 25 min
```

Zadatak 1.1.8 Napisati program koji razmenjuje vrednosti dveju promenljivih x i y. Njihove vrednosti, kao dva cela broja, zadaje korisnik.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vrednosti x i y: 5 7
 Pre zamene: x = 5, y = 7
 Posle zamene: x = 7, y = 5
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vrednosti x i y: 237 -592
 Pre zamene: x = 237, y = -592
 Posle zamene: x = -592, y = 237
```

Zadatak 1.1.9 Date su dve celobrojene promenljive a i b. Napisati program koji promenljivoj a dodeljuje njihovu sumu, a promenljivoj b njihovu razliku. NAPOMENA: *Ne koristiti pomoćne promenljive.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vrednosti a i b: 5 7
 Nove vrednosti su: a = 12, b = -2
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite vrednosti a i b: 237 -592
 Nove vrednosti su: a = -355, b = 829
```

Zadatak 1.1.10 Napisati program koji za uneti pozitivan trocifreni broj ispisuje njegove cifre jedinica, desetica i stotina. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite trocifreni broj: 697
 Cifra jedinica: 7
 Cifra desetica: 9
 Cifra stotina: 6
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite trocifreni broj: 504
 Cifra jedinica: 4
 Cifra desetica: 0
 Cifra stotina: 5
```

Zadatak 1.1.11 Napisati program koji za unetu cenu proizvoda ispisuje najmanji broj novčanica koje je potrebno izdvojiti prilikom plaćanja proizvoda. Na raspolaganju su novčanice od 5000, 2000, 1000, 500, 200, 100, 50, 20, 10 i

1 dinar. Cena proizvoda je pozitivan ceo broj. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite cenu proizvoda: 8367
 8367 = 1*5000 + 1*2000 + 1*1000 + 0*500 + 1*200 + 1*100 + 1*50 + 0*20 + 1*10 + 7*1
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite cenu proizvoda: 934
 934 = 0*5000 + 0*2000 + 0*1000 + 1*500 + 2*200 + 0*100 + 0*50 + 1*20 + 1*10 + 4*1
```

Zadatak 1.1.12 Napisati program koji učitava pozitivan trocifreni broj i ispisuje broj dobijen obrtanjem njegovih cifara. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite trocifreni broj: 892
 Obrnuto: 298
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite trocifreni broj: 230
 Obrnuto: 32
```

Zadatak 1.1.13 Napisati program koji za uneti pozitivan četvorocifreni broj:

(a) izračunava proizvod cifara

(b) izračunava razliku sume krajnjih i srednjih cifara

(c) izračunava sumu kvadrata cifara

(d) izračunava broj koji se dobija zapisom cifara u obrnutom poretku

(e) izračunava broj koji se dobija zamenom cifre jedinice i cifre stotine

NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite cetvorocifreni broj: 2371
 Proizvod cifara: 42
 Razlika sume krajnjih i srednjih: -7
 Suma kvadrata cifara: 63
 Broj u obrnutom poretku: 1732
 Broj sa zamenjenom cifrom
 jedinica i stotina: 2173
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite cetvorocifreni broj: 3570
 Proizvod cifara: 0
 Razlika sume krajnjih i srednjih: -9
 Suma kvadrata cifara: 83
 Broj sa zamenjenom cifrom
 jedinica i stotina: 3075
```

Zadatak 1.1.14 Napisati program koji ispisuje broj koji se dobija izbacivanjem cifre desetica u unetom pozitivnom celom broju. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj: 1349
 Rezultat: 139
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj: 825
 Rezultat: 85
```

Zadatak 1.1.15 Napisati program koji učitava pozitivan ceo broj n i pozitivan dvocifreni broj m i ispisuje broj dobijen umetanjem broja m između cifre stotina i cifre hiljada broja n. NAPOMENA: *Za neke ulazne podatke može se dobiti neočekivan rezultat zbog prekoračenja, što ilustruje test primer broj 2.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite pozitivan ceo broj: 12345
 Unesite pozitivan dvocifreni broj: 67
 Rezultat: 1267345
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite pozitivan ceo broj: 50000000
 Unesite pozitivan dvocifreni broj: 12
 Rezultat: 705044704
```

Zadatak 1.1.16 Napisati program koji učitava dužinu dijagonale monitora izraženu u inčima, konvertuje je u centimetre i ispisuje zaokruženu na dve decimale. UPUTSTVO: *Jedan inč ima* $2,54$ *centimetra.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj inca: 4.69
 4.69 in = 11.91 cm
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj inca: 71.426
 71.43 in = 181.42 cm
```

Zadatak 1.1.17 Napisati program koji učitava dužinu pređenog puta izraženu u miljama, konvertuje je u kilometre i ispisuje zaokruženu na dve decimale. UPUTSTVO: *Jedna milja ima* $1,609344$ *kilometara.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj milja: 50.42
 50.42 mi = 81.14 km
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj milja: 327.128
 327.128 mi = 526.46 km
```

Zadatak 1.1.18 Napisati program koji učitava težinu avionskog tereta izraženu u funtama, konvertuje je u kilograme i ispisuje zaokruženu na dve decimale. UPUTSTVO: *Jedna funta ima* $0,45359237$ *kilograma.*

Primer 1

```
Interakcija sa programom:
 Unesite broj funti: 2.78
 2.78 lb = 1.26 kg
```

Primer 2

```
Interakcija sa programom:
 Unesite broj funti: 89.437
 89.437 lb = 40.57 kg
```

Zadatak 1.1.19 Napisati program koji učitava temperaturu izraženu u farenhajtima, konvertuje tu vrednost u celzijuse i ispisuje je zaokruženu na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.* Uputstvo: *Veza između farenhajta i celzijusa je zadata narednom formulom* $F = \frac{9 \cdot C}{5} + 32$

Primer 1

```
Interakcija sa programom:
 Unesite temperaturu u F: 100.93
 100.93 F = 38.29 C
```

Primer 2

```
Interakcija sa programom:
 Unesite temperaturu u F: 25.562
 25.562 F = -3.58 C
```

Zadatak 1.1.20 Napisati program koji za unete realne vrednosti a_{11}, a_{12}, a_{21}, a_{22} ispisuje vrednost determinante matrice:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

Pri ispisu vrednost zaokružiti na četiri decimale.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve:
 1 2 3 4
 Determinanta: -2.0000
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve:
 1.5 -2 3 4.5
 Determinanta: 12.7500
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve:
 0.01 0.01 0.5 7
 Determinanta: 0.0650
```

Zadatak 1.1.21 Napisati program koji za unete realne vrednosti dužina stranica pravougaonika ispisuje njegov obim i površinu. Ispisati tražene vrednosti zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite duzine stranica: 4.3 9.4
 Obim: 27.40
 Povrsina: 40.42
```

Primer 2

```
Interakcija sa programom:
 Unesite duzine stranica: 10.756 36.2
 Obim: 93.91
 Povrsina: 389.37
```

Zadatak 1.1.22 Napisati program koji za unetu realnu vrednost dužine poluprečnika kruga ispisuje njegov obim i površinu zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite poluprecnik: 4.2
 Obim: 26.39
 Povrsina: 55.42
```

Primer 2

```
Interakcija sa programom:
 Unesite poluprecnik: 14.932
 Obim: 93.82
 Povrsina: 700.46
```

Zadatak 1.1.23 Napisati program koji za unetu realnu vrednost dužine stranice jednakostraničnog trougla ispisuje njegov obim i površinu zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.* Uputstvo: *Za računanje površine jednakostraničnog trougla može se iskoristiti obrazac* $P = \frac{a^2 \cdot \sqrt{3}}{4}$ *pri čemu je* a *dužina stranice. Za računanje korena broja koristiti funkciju* **`sqrt`** *čija se deklaracija nalazi u zaglavlju* **`math.h`**.

Primer 1

```
Interakcija sa programom:
 Unesite duzinu stranice trougla: 5
 Obim: 15.00
 Povrsina: 10.82
```

Primer 2

```
Interakcija sa programom:
 Unesite duzinu stranice trougla: 2
 Obim: 6.00
 Povrsina: 1.73
```

Zadatak 1.1.24 Napisati program koji za unete realne vrednosti dužina stranica trougla ispisuje njegov obim i površinu zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.* Uputstvo: *Za računanje površine trougla može se koristiti Heronov obrazac* $P = \sqrt{S \cdot (S-a) \cdot (S-b) \cdot (S-c)}$, *pri čemu su* a, b *i* c *dužine stranica, a* S *je poluobim.*

Primer 1

```
Interakcija sa programom:
 Unesite duzine stranica trougla:
 3 4 5
 Obim: 12.00
 Povrsina: 6.00
```

Primer 2

```
Interakcija sa programom:
 Unesite duzine stranica trougla:
 4.3 9.7 8.8
 Obim: 22.80
 Povrsina: 18.91
```

Zadatak 1.1.25 Pravougaonik čije su stranice paralelne koordinatnim osama zadat je svojim realnim koordinatama naspramnih temena (gornje levo i donje desno teme). Napisati program koji ispisuje njegov obim i površinu zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite koordinate gornjeg levog temena: 4.3 5.8
 Unesite koordinate donjeg desnog temena: 6.7 2.3
 Obim: 11.80
 Povrsina: 8.40
```

Primer 2

```
Interakcija sa programom:
 Unesite koordinate gornjeg levog temena: -3.7 8.23
 Unesite koordinate donjeg desnog temena: -0.56 2
 Obim: 18.74
 Povrsina: 19.56
```

Zadatak 1.1.26 Napisati program koji za tri uneta cela broja ispisuje njihovu artimetičku sredinu zaokruženu na dve decimale.

Primer 1

```
Interakcija sa programom:
 Unesite tri cela broja: 11 5 4
 Aritmeticka sredina: 6.67
```

Primer 2

```
Interakcija sa programom:
 Unesite tri cela broja: 3 -8 13
 Aritmeticka sredina: 2.67
```

Zadatak 1.1.27 Napisati program koji pomaže moleru da izračuna površinu zidova prostorije koju treba da okreči. Za unete celobrojne vrednosti dimenzije sobe u metrima (dužinu, širinu i visinu), program treba da ispiše površinu zidova za krečenje pod pretpostavkom da na vrata i prozore otpada oko 20%. Omogućiti i da na osnovu unete celobrojene cene usluge po kvadratnom metru program izračuna ukupnu cenu krečenja. Sve realne vrednosti ispisati zaokružene na dve decimale. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite dimenzije sobe: 4 4 3
 Unesite cenu po m2: 500
 Moler treba da okreci 51.20 m2
 Cena krecenja: 25600.00
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenzije sobe: 13 17 3
 Unesite cenu po m2: 475
 Moler treba da okreci 320.80 m2
 Cena krecenja: 152380.00
```

Zadatak 1.1.28 Napisati program koji za unete pozitivne cele brojeve x, p i c ispisuje broj koji se dobija ubacivanjem cifre c u broj x na poziciju p. Pretpostaviti da numeracija cifara počinje od nule, odnosno da se cifra najmanje težine nalazi se na nultoj poziciji. Napomena: *Pretpostaviti da je unos ispravan.* Uputstvo: *Koristiti funkciju* `pow` *čija se deklaracija nalazi u zaglavlju* `math.h`.

Primer 1

```
Interakcija sa programom:
 Unesite redom x, p i c: 140 1 2
 Rezultat: 1420
```

Primer 1

```
Interakcija sa programom:
 Unesite redom x, p i c: 12345 2 9
 Rezultat: 123945
```

Zadatak 1.1.29 Napisati program koji za uneta dva cela broja a i b dodeljuje promenljivoj *rezultat* vrednost 1 ako važi uslov:

(a) a i b su različiti brojevi

(b) a i b su parni brojevi

(c) a i b su pozitivni brojevi, ne veći od 100

U suprotnom, promenljivoj *rezultat* dodeliti vrednost 0. Ispisati vrednost promenljive *rezultat*.

Primer 1

```
Interakcija sa programom:
 Unesite dva cela broja: 4 8
 a) Rezultat: 1
 b) Rezultat: 1
 c) Rezultat: 1
```

Primer 2

```
Interakcija sa programom:
 Unesite dva cela broja: 3 -11
 a) Rezultat: 1
 b) Rezultat: 0
 c) Rezultat: 0
```

Zadatak 1.1.30 Napisati program koji za uneta dva cela broja ispisuje njihov maksimum.

Primer 1

```
Interakcija sa programom:
 Unesite dva cela broja: 19 256
 Maksimum: 256
```

Primer 2

```
Interakcija sa programom:
 Unesite dva cela broja: -39 57
 Maksimum: 57
```

Zadatak 1.1.31 Napisati program koji za uneta dva cela broja ispisuje njihov minimum.

Primer 1

```
Interakcija sa programom:
 Unesite dva cela broja: 4 8
 Minimum: 4
```

Primer 2

```
Interakcija sa programom:
 Unesite dva cela broja: -3 -110
 Minimum: -110
```

Zadatak 1.1.32 Napisati program koji za zadate realne vrednosti x i y ispisuje vrednost sledećeg izraza

$$rez = \frac{\min(x, y) + 0.5}{1 + \max^2(x, y)}$$

zaokruženu na dve decimale.

Primer 1

```
Interakcija sa programom:
 Unesite dva realna broja: 5.7 11.2
 Rezultat: 0.05
```

Primer 1

```
Interakcija sa programom:
 Unesite dva realna broja: -9.34 8.99
 Rezultat: -0.11
```

1.2 Rešenja

Rešenje 1.1.1

```
#include <stdio.h>

/* Prevodjenje programa program.c na Linux operativnom sistemu
  koriscenjem kompajlera gcc vrsi se iz terminala komandom
  gcc program.c
  Ukoliko program.c ne sadrzi greske, kompajler ce napraviti
  izvrsnu verziju programa koja ce se zvati a.out
  Ovaj program se moze pokrenuti iz terminala navodjenjem komande
  ./a.out
  Ukoliko se program prevede na sledeci nacin
  gcc program.c -o program
  onda ce izvrsna verzija biti imenovana nazivom koji je dat nakon
  -o opcije (u ovom slucaju je to rec program) tako da se pokretanje
  iz terminala moze uraditi navodjenjem komande
  ./program
*/

int main() {
  /* Ispis trazene poruke. Na kraju poruke se ispisuje novi red. */
  printf("Zdravo svima!\n");

  /* Povratna vrednost 0 se obicno koristi da oznaci da je prilikom
     izvrsavanja programa sve proslo u redu. */
  return 0;
}
```

Rešenje 1.1.2

```
#include <stdio.h>

int main() {
  /* Deklaracija celobrojne promenljive. */
  int n;

  /* Ucitavanje vrednosti celog broja. */
  printf("Unesite ceo broj: ");
  scanf("%d", &n);

  /* Ispis kvadratne vrednosti unetog broja. */
  printf("Kvadrat: %d\n", n * n);

  /* Ispis kubne vrednosti unetog broja. */
  printf("Kub: %d\n", n * n * n);

  return 0;
}
```

Rešenje 1.1.3

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x, y, rezultat;

  /* Ucitavanje vrednosti broja x. */
  printf("Unesite vrednost broja x: ");
  scanf("%d", &x);

  /* Ucitavanje vrednosti broja y. */
  printf("Unesite vrednost broja y: ");
  scanf("%d", &y);

  /* I nacin ispisa: Dodela zbira x+y promenljivoj rezultat i
     ispis vrednosti promenljive rezultat. */
  rezultat = x + y;
  printf("%d + %d = %d\n", x, y, rezultat);

  /* II nacin ispisa: Direktan ispis vrednosti izraza, bez njegovog
     dodeljivanja posebnoj promenljivoj. */
  printf("%d - %d = %d\n", x, y, x - y);
  printf("%d * %d = %d\n", x, y, x * y);

  /* Kada se operator / primeni na dva celobrojna operanda x i y,
     kao rezultat se dobije ceo deo pri deljenju broja x brojem y,
     a ne kolicnik. Na primer, rezultat primene operatora / na 7 i 2
     je 3, a ne 3.5. */
  printf("%d / %d = %d\n", x, y, x / y);

  /* Operator % izracunava ostatak pri celobrojnom deljenju dve
     celobrojne promenljive ili izraza.
     Da bi se odstampao karakter %, u pozivu funkcije printf se pise
    %%. */
  printf("%d %% %d = %d\n", x, y, x % y);

  return 0;
}
```

Rešenje 1.1.4 Pogledajte zadatak 1.1.3. Zbog pretpostavke da su cene artikala pozitivni celi brojevi, tip promenljivih za cene treba da bude `unsigned int`.

Rešenje 1.1.5 Pogledajte zadatak 1.1.3. Zbog pretpostavke da su cene artikala pozitivni celi brojevi, tip promenljivih za cene treba da bude `unsigned int`.

Rešenje 1.1.6

```
#include <stdio.h>

int main() {
  /* Deklaracija promenljivih cija je vrednost neoznacen ceo broj. */
  unsigned int cena, kolicina, iznos, kusur;

  /* Ucitavanje vrednosti cene, kolicine i iznosa. */
  printf("Unesite cenu, kolicinu i iznos:\n");
  scanf("%u%u%u", &cena, &kolicina, &iznos);

  /* Racunanje kusura. */
  kusur = iznos - kolicina * cena;

  /* Ispis vrednosti kusura. */
  printf("Kusur: %u dinara\n", kusur);

  return 0;
}
```

Rešenje 1.1.7

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int poletanje, poletanje_sat, poletanje_minut;
  unsigned int sletanje, sletanje_sat, sletanje_minut;
  unsigned int duzina, duzina_sat, duzina_minut;

  /* Ucitavanje vremena poletanja. */
  printf("Unesite vreme poletanja: ");
  scanf("%u%u", &poletanje_sat, &poletanje_minut);

  /* Ucitavanje vremena sletanja. */
  printf("Unesite vreme sletanja: ");
  scanf("%u%u", &sletanje_sat, &sletanje_minut);

  /* Pretvaranje oba vremena u minute radi lakseg racunanja
     razlike. */
  poletanje = poletanje_sat * 60 + poletanje_minut;
  sletanje = sletanje_sat * 60 + sletanje_minut;

  /* Racunanje razlike u minutima izmedju sletanja i poletanja. */
  duzina = sletanje - poletanje;

  /* Pretvaranje razlike u minutama u razliku u satima i minutima.
     Razlika u satima se dobija celobrojnim deljenjem broja minuta
     sa 60. Preostali broj minuta se dobija kao ostatak pri
     deljenju sa 60. */
```

```
  duzina_sat = duzina / 60;
  duzina_minut = duzina % 60;

  /* II nacin: duzina_minut = duzina - duzina * 60; */

  /* Ispis rezultata. */
  printf("Duzina trajanja leta: %u h i %u min\n", duzina_sat,
         duzina_minut);

  return 0;
}
```

Rešenje 1.1.8

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x, y, p;

  /* Ucitavanje vrednosti x i y. */
  printf("Unesite vrednosti x i y: ");
  scanf("%d%d", &x, &y);

  /* Ispis vrednosti promenljivih pre zamene. */
  printf("Pre zamene: x=%d, y=%d\n", x, y);

  /* Pomocna promenljiva p je potrebna da sacuva vrednost
     promenljive x pre nego sto se ona izmeni i dobije vrednost
     promenljive y. */
  p = x;
  x = y;
  y = p;

  /* Ispis vrednosti promenljivih nakon zamene. */
  printf("Posle zamene: x=%d, y=%d\n", x, y);

  return 0;
}
```

Rešenje 1.1.9

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a, b;

  /* Ucitavanje vrednosti a i b. */
  printf("Unesite vrednosti a i b: ");
```

```
  scanf("%d%d", &a, &b);

  /* Smestanje sume a + b u promenljivu a. */
  a = a + b;

  /* Smestanje izraza a - 2*b u promenljivu b. Uzimajuci u obzir
     promenu vrednosti promenljive a, u odnosu na pocetne vrednosti
     promenljivih a i b, vrednost ovog izraza je jednaka
     a + b - 2*b = a - b. */
  b = a - 2 * b;

  /* Ispis rezultata. */
  printf("Nove vrednosti su: a=%d, b=%d\n", a, b);

  return 0;
}
```

Rešenje 1.1.10

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebne promenljive. */
  unsigned int x;

  /* Promenljive koje cuvaju cifre treba da budu najmanjeg
     celobrojnog tipa jer nece sadrzati druge vrednosti osim
     jednocifrenih celih brojeva. Zbog toga se koristi tip char. */
  char cifra_jedinica, cifra_desetica, cifra_stotina;

  /* Ucitavanje trocifrenog broja. */
  printf("Unesite trocifreni broj: ");
  scanf("%u", &x);

  /* Izdvajanje cifara jedinice, desetice i stotine. */
  cifra_jedinica = x % 10;
  cifra_desetica = (x / 10) % 10;
  cifra_stotina = x / 100;

  /* Ispis rezultata.
     Napomena: Kada se stampa numericka vrednost promenljive tipa
     char koristi se %d. Kada se stampa karakter ciji je ASCII
     kod jednak vrednosti te promenljive, tada se koristi %c.
     U ovom slucaju je potrebno stampati numericku vrednost. */
  printf("Cifra jedinica: %d\n", cifra_jedinica);
  printf("Cifra desetica: %d\n", cifra_desetica);
  printf("Cifra stotina: %d\n", cifra_stotina);

  /* II nacin: Ispis rezultata bez uvodjenja dodatnih promenljivih
     cifra_jedinica, cifra_desetica i cifra_stotina:
     printf("Cifre unetog broja su %d,%d,%d\n", x%10, (x/10)%10,
```

```
33              x/100); */

35    return 0;
  }
```

Rešenje 1.1.11

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracija potrebne promenljive. */
5    unsigned int cena;

7    /* Ucitavanje cene proizvoda. */
     printf("Unesite cenu proizvoda: ");
9    scanf("%u", &cena);

11   /* Vrednost cena/5000 predstavlja maksimalan broj novcanica od 5000
        dinara koje je moguce iskoristiti za placanje racuna.
13      Na primer, neka je uneta cena 8367 dinara, vrednost izraza
        8367/5000 je jednaka 1. */
15   printf("%u = %u*5000 + ", cena, cena / 5000);

17   /* Da bi se isti postupak primenio i na ostale novcanice, potrebno
        je izracunati preostali iznos. Jedan nacin da se to uradi je
19      racunanje ostatka pri deljenju unete vrednosti cena
        (u primeru 8367) sa 5000. On iznosi 3367. Ova vrednost se
21      dodeljuje promeljivoj cena. */
     cena = cena % 5000;
23
     /* Ponavljanje postupka za ostale novcanice. */
25   printf("%u*2000 + ", cena / 2000);
     cena = cena % 2000;
27   printf("%u*1000 + ", cena / 1000);
     cena = cena % 1000;
29   printf("%u*500 + ", cena / 500);
     cena = cena % 500;
31   printf("%u*200 + ", cena / 200);
     cena = cena % 200;
33   printf("%u*100 + ", cena / 100);
     cena = cena % 100;
35   printf("%u*50 + ", cena / 50);
     cena = cena % 50;
37   printf("%u*20 + ", cena / 20);
     cena = cena % 20;
39   printf("%u*10 + ", cena / 10);
     cena = cena % 10;
41   printf("%u*1\n", cena);

43   return 0;
  }
```

Rešenje 1.1.12

```c
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int x, obrnuto_x;
  char cifra_jedinice, cifra_desetice, cifra_stotine;

  /* Ucitavanje neoznacenog trocifrenog broja. */
  printf("Unesite trocifreni broj: ");
  scanf("%u", &x);

  /* Izdvajanje pojedinacnih cifara broja. */
  cifra_jedinice = x % 10;
  cifra_desetice = (x / 10) % 10;
  cifra_stotine = x / 100;

  /* Formiranje rezultujuceg broja. */
  obrnuto_x = cifra_jedinice * 100 + cifra_desetice * 10 +
    cifra_stotine;

  /* Ispis rezultata. */
  printf("Obrnuto: %u\n", obrnuto_x);

  return 0;
}
```

Rešenje 1.1.13 Pogledajte zadatak 1.1.12.

Rešenje 1.1.14

```c
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int broj, novi_broj;
  unsigned int levo, desno;

  /* Ucitavanje neoznacenog celog broja. */
  printf("Unesite broj: ");
  scanf("%u", &broj);

  /* Desni deo rezultata je cifra jedinice unetog broja.
     Na primer, za broj 1234, desni deo je cifra 4. */
  desno = broj%10;

  /* Levi deo rezultata su sve cifre levo od cifre desetice.
     Na primer, za broj 1234, levi deo je broj 12 i dobija se
     deljenjem unetog broja sa 100. */
  levo = broj/100;
```

```
21   /* Rezultat se dobija spajanjem levog i desnog dela.
        U datom primeru: 12*10 + 4 = 124. */
23   novi_broj = levo*10 + desno;

25   /* Ispis rezultata. */
     printf("Rezultat: %u\n", novi_broj);
27
     return 0;
29 }
```

Rešenje 1.1.15

```
   #include <stdio.h>
2
   int main() {
4    /* Deklaracije potrebnih promenljivih. */
     unsigned int n, novi_broj;
6    unsigned int levi, desni, m;

8    /* Ucitavanje brojeva n i m. */
     printf("Unesite pozitivan ceo broj: ");
10   scanf("%u", &n);
     printf("Unesite pozitivan dvocifreni broj: ");
12   scanf("%u", &m);

14   /* Levi deo rezultata su sve cifre levo od cifre stotina.
        Na primer, ako je n=12345, levi deo rezultata je 12.
16      On se dobija deljenjem unetog broja sa 1000. */
     levi = n / 1000;
18
     /* Desni deo rezultata su sve cifre desno od cifre hiljada.
20      Za n=12345, desni deo rezultata je 345. */
     desni = n % 1000;
22
     /* Srednji deo rezultata je broj m.
24      U navedenom primeru, rezultat se dobija nadovezivanjem
        brojeva 12, 67 i 345. Ovo se radi mnozenjem delova
26      odgovarajucim stepenom broja 10 i njihovim sabiranjem. */
     novi_broj = levi * 100000 + m * 1000 + desni;
28
     /* Ispis rezultata. */
30   printf("Rezultat: %u\n", novi_broj);

32   return 0;
   }
```

Rešenje 1.1.16

```c
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float in, cm;

  /* Ucitavanje realnog broja koji predstavlja broj inca. */
  printf("Unesite broj inca: ");
  scanf("%f", &in);

  /* Racunanje rezultata (1 in = 2.54 cm) */
  cm = in * 2.54;

  /* Ispis rezultata (na dve decimale). */
  printf("%.2f in = %.2f cm\n", in, cm);

  return 0;
}
```

Rešenje 1.1.17 Pogledajte zadatak 1.1.16.

Rešenje 1.1.18 Pogledajte zadatak 1.1.16.

Rešenje 1.1.19 Pogledajte zadatak 1.1.16.

Rešenje 1.1.20

```c
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float a11, a12, a21, a22, determinanta;

  /* Ucitavanje elemenata matrice. */
  printf("Unesite brojeve: ");
  scanf("%f%f%f%f", &a11, &a12, &a21, &a22);

  /* Racunanje determinante matrice. */
  determinanta = a11*a22 - a12*a21;

  /* Ispis rezultata na cetiri decimale. */
  printf("Determinanta: %.4f\n", determinanta);

  return 0;
}
```

Rešenje 1.1.21

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float a, b, obim, povrsina;

  /* Ucitavanje duzina stranica pravougaonika. */
  printf("Unesite duzine stranica pravougaonika: ");
  scanf("%f%f", &a, &b);

  /* Racunanje obima pravougaonika. */
  obim = 2 * (a + b);

  /* Racunanje povrsine pravougaonika. */
  povrsina = a * b;

  /* Ispis rezultata na dve decimale. */
  printf("Obim: %.2f\n", obim);
  printf("Povrsina: %.2f\n", povrsina);

  return 0;
}
```

Rešenje 1.1.22

```
#include <stdio.h>
#include <math.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float r, obim, povrsina;

  /* Ucitavanje poluprecnika kruga. */
  printf("Unesite poluprecnik: ");
  scanf("%f", &r);

  /* Racunanje obima i povrsine.
     M_PI je konstanta koja se nalazi u zaglavlju math.h
     i njena vrednost odgovara pribliznoj vrednosti broja pi. */
  obim = 2 * r * M_PI;
  povrsina = r * r * M_PI;

  /* Ispis rezultata na dve decimale. */
  printf("Obim: %.2f\nPovrsina: %.2f\n", obim, povrsina);

  return 0;
}
```

Rešenje 1.1.23

```
#include <stdio.h>
#include <math.h>

/* Uvek kada se koristi neka funkcija iz matematicke bibilioteke
   (na primer , funkcije sqrt , pow , sin , cos)
   potrebno je prilikom prevodjenja dodati i opciju -lm kojom se
   kompajler upucuje da je za pravljenje izvrsne verzije programa
   potrebno da se program poveze sa matematickom bibliotekom.
   Ukoliko se ova opcija ne navede, kompajler prijavljuje gresku:
   collect2: error: ld returned 1 exit status
*/

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float a, povrsina, obim;

  /* Ucitavanje duzina stranice. */
  printf("Unesite duzinu stranice trougla: ");
  scanf("%f", &a);

  /* Racunanje obima i povrsine. */
  obim = 3 * a;
  povrsina = (a * a * sqrt(3)) / 4;

  /* Ispis rezultata na dve decimale. */
  printf("Obim: %.2f\n", obim);
  printf("Povrsina: %.2f\n", povrsina);

  return 0;
}
```

Rešenje 1.1.24 Pogledajte zadatke 1.1.21 i 1.1.22.

Rešenje 1.1.25 Pogledajte zadatak 1.1.21.

Rešenje 1.1.26

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a, b, c;
  float aritmeticka_sredina;

  /* Ucitavanje vrednosti tri cela broja. */
  printf("Unesite tri cela broja:");
  scanf("%d%d%d", &a, &b, &c);

  /* Pogresan nacin: aritmeticka_sredina = (a+b+c)/3;
```

```
13      Kada se operacija / koristi nad celim brojevima,
        deljenje je celobrojno.
15      Na primer, (1+1+3)/3 ima vrednost 1.*/

17    /* Ispravan nacin je da se bar jedan operand
         pretvori u realan broj. */
19    aritmeticka_sredina = (a + b + c) / 3.0;

21    /* Drugi ispravni nacini:
         aritmeticka_sredina=1.0*(a+b+c)/3;
23       aritmeticka_sredina=(0.0+a+b+c)/3;
         aritmeticka_sredina=((float)(a+b+c))/3; */
25
      /* Ispis rezultata. */
27    printf("Aritmeticka sredina:  %.2f\n", aritmeticka_sredina);

29    return 0;
  }
```

Rešenje 1.1.27

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracije potrebnih promenljivih. */
5    unsigned int duzina, sirina, visina, cena;
     float povrsina_za_krecenje, ukupna_cena;
7
     /* Ucitavanje vrednosti duzine, sirine i visine sobe. */
9    printf("Unesite dimenzije sobe: ");
     scanf("%u%u%u", &duzina, &sirina, &visina);
11
     /* Ucitavanje vrednosti cene krecenja. */
13   printf("Unesite cenu po m2: ");
     scanf("%u", &cena);
15
     /* Povrsina za krecenje odgovara povrsini kvadra
17      umanjena za povrsinu poda jer se on ne kreci. */
     povrsina_za_krecenje = 0.8 * (duzina * sirina +
19     2 * duzina * visina + 2 * sirina * visina);

21   /* Racunanje ukupne cene. */
     ukupna_cena = povrsina_za_krecenje * cena;
23
     /* Ispis rezultata. */
25   printf("Moler treba da okreci %.2f m2\n", povrsina_za_krecenje);
     printf("Cena krecenja: %.2f\n", ukupna_cena);
27
     return 0;
29 }
```

Rešenje 1.1.28

```
#include <stdio.h>
#include <math.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int x, p, c;
  unsigned int levo, desno, novo_x;

  /* Ucitavanje broja, pozicije i cifre. */
  printf("Unesite redom x, p i c: ");
  scanf("%u%u%u", &x, &p, &c);

  /* Racunanje dela broja koji se nalazi desno od pozicije p.
     Funkcija pow kao povratnu vrednost vraca realan broj dvostruke
     tacnosti, a operacija % ocekuje celobrojne operande. Iz tog
     razloga je neophodno izvrsiti pretvaranje povratne vrednosti
     u tip unsigned int. */
  desno = x % (unsigned int) pow(10, p);

  /* Racunanje dela broja koji se nalazi levo od pozicije p. */
  levo = x / (unsigned int) pow(10, p);

  /* Racunanje rezultata nadovezivanjem levog dela, cifre c
     i desnog dela. */
  novo_x =levo * (unsigned int) pow(10, p + 1) +
         c * (unsigned int) pow(10, p) + desno;

  /* Ispis rezultata. */
  printf("Rezultat: %u\n", novo_x);

  return 0;
}
```

Rešenje 1.1.29

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a, b;
  int rezultat_a, rezultat_b, rezultat_c;

  /* Ucitavanje dva cela broja. */
  printf("Unesite dva cela broja: ");
  scanf("%d%d", &a, &b);

  /* Izraz a != b ima vrednost 1 ako je ova relacija tacna, a 0 ako
     je netacna. */
  rezultat_a = a != b;
```

```
15
   /* Izraz a % 2 == 0 && b % 2 == 0 je konjunkcija koja se sastoji
17    od dve relacije poredjenja jednakosti. Izraz a % 2 == 0 ima
      vrednost 1 ako je ova relacija tacna, a 0 u suprotnom. */
19 rezultat_b = (a % 2 == 0 && b % 2 == 0);

21 /* Izraz a > 0 && a <= 100 && b > 0 && b <= 100 je konjunkcija
      koja se sastoji od cetiri konjunkta. Svaki od konjunkata je
23    izraz koji sadrzi relacioni operator i ima vrednost 1 ako
      relacija vazi, a 0 ako ne vazi. */
25 rezultat_c = (a > 0 && a <= 100 && b > 0 && b <= 100);

27 /* Ispis rezultata. */
   printf("a) Rezultat: %d\n", rezultat_a);
29 printf("b) Rezultat: %d\n", rezultat_b);
   printf("c) Rezultat: %d\n", rezultat_c);
31
   return 0;
33 }
```

Rešenje 1.1.30

```
1 #include <stdio.h>

3 int main() {
   /* Deklaracija potrebnih promenljivih. */
5  int a, b, max;

7  /* Ucitavanje dve celobrojne vrednosti. */
   printf("Unesite dva cela broja: ");
9  scanf("%d%d", &a, &b);

11 /* Racunanje maksimuma koriscenjem ternarnog operatora. */
   max = (a > b) ? a : b;
13
   /* Ispis rezultata. */
15 printf("Maksimum: %d\n", max);

17 return 0;
}
```

Rešenje 1.1.31 Pogledajte zadatak 1.1.30

Rešenje 1.1.32

```
1 #include <stdio.h>

3 int main() {
   /* Deklaracije potrebnih promenljivih. */
5  float a, b, rezultat;
```

```
    float min, max;
7
    /* Ucitavanje vrednosti dva realna broja. */
9   printf("Unesite dva realna broja: ");
    scanf("%f%f", &a, &b);
11
    /* Racunanje minimalne i maksimalne vrednost unetih brojeva. */
13  min = (a < b) ? a : b;
    max = (a > b) ? a : b;
15
    /* Racunanje rezultata. */
17  rezultat = (min + 0.5) / (1 + max * max);

19  /* Ispis rezultata. */
    printf("Rezultat: %.2f\n", rezultat);
21
    return 0;
23 }
```

1.3 Grananja

Zadatak 1.3.1 Napisati program koji ispisuje najmanji od tri uneta cela broja.

Primer 1

```
Interakcija sa programom:
 Unesite tri cela broja:
 5 18 -1
 Najmanji: -1
```

Primer 2

```
Interakcija sa programom:
 Unesite tri cela broja:
 0 43 16
 Najmanji: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite tri cela broja:
 -5 -5 -5
 Najmanji: -5
```

Zadatak 1.3.2 Napisati program koji za uneti realan broj ispisuje njegovu apsolutnu vrednost zaokruženu na dve decimale.

Primer 1

```
Interakcija sa programom:
 Unesite jedan realan broj:
 7.42
 Apsolutna vrednost: 7.42
```

Primer 2

```
Interakcija sa programom:
 Unesite jedan realan broj:
 -562.428
 Apsolutna vrednost: 562.43
```

Primer 3

```
Interakcija sa programom:
 Unesite jedan realan broj:
 0
 Apsolutna vrednost: 0.00
```

Zadatak 1.3.3 Napisati program koji za uneti ceo broj ispisuje njegovu recipročnu vrednost zaokruženu na četiri decimale. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite jedan ceo broj: 22
 Reciprocna vrednost: 0.0455
```

Primer 2

```
Interakcija sa programom:
 Unesite jedan ceo broj: -9
 Reciprocna vrednost: -0.1111
```

Primer 3

```
Interakcija sa programom:
 Unesite jedan ceo broj: 0
 Greska: nedozvoljeno je deljenje nulom.
```

Primer 4

```
Interakcija sa programom:
 Unesite jedan ceo broj: 57298
 Reciprocna vrednost: 0.0000
```

Zadatak 1.3.4 Napisati program koji učitava tri cela broja i ispisuje zbir pozitivnih.

Primer 1

```
Interakcija sa programom:
 Unesite tri cela broja:
 1 3 -6
 Zbir pozitivnih: 4
```

Primer 2

```
Interakcija sa programom:
 Unesite tri cela broja:
 -719 -48 -123
 Zbir pozitivnih: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite tri cela broja:
 16 2 576
 Zbir pozitivnih: 594
```

Zadatak 1.3.5 U prodavnici je organizovana akcija da svaki kupac dobije najjeftiniji od tri artikla za jedan dinar. Napisati program koji za unete cene tri artikla izračunava ukupnu cenu, kao i koliko dinara se uštedi zahvaljujući popustu. Cene artikala su pozitivni celi brojevi. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite tri cene: 35 125 97
 Cena sa popustom: 223 din
 Usteda: 34 din
```

Primer 2

```
Interakcija sa programom:
 Unesite tri cene: 1034 15 25
 Cena sa popustom: 1060 din
 Usteda: 14 din
```

Primer 3

```
Interakcija sa programom:
 Unesite tri cene: 500 500 500
 Cena sa popustom: 1001 din
 Usteda: 499 din
```

Primer 4

```
Interakcija sa programom:
 Unesite tri cene: 247 -133 126
 Greska: neispravan unos cene.
```

Zadatak 1.3.6 Napisati program koji za uneto vreme u formatu *sat:minut* ispisuje koliko je sati i minuta ostalo do ponoći. Broj sati treba da bude iz intervala $[0, 24)$, a broj minuta iz intervala $[0, 60)$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite vreme: 18:19
 Do ponoci: 5 sati i 41 minuta
```

Primer 2

```
Interakcija sa programom:
 Unesite vreme: 23:7
 Do ponoci: 0 sati i 53 minuta
```

Primer 3

```
Interakcija sa programom:
 Unesite vreme: 24:20
 Greska: neispravan unos vremena.
```

Primer 4

```
Interakcija sa programom:
 Unesite vreme: 14:0
 Do ponoci: 10 sati i 0 minuta
```

Zadatak 1.3.7 Napisati program koji za unetu godinu ispisuje da li je prestupna. Godina je neoznačen ceo broj.

Primer 1

```
Interakcija sa programom:
 Unesite godinu: 2016
 Godina je prestupna.
```

Primer 2

```
Interakcija sa programom:
 Unesite godinu: 1997
 Godina nije prestupna.
```

Primer 3

```
Interakcija sa programom:
 Unesite godinu: 1900
 Godina nije prestupna.
```

Zadatak 1.3.8 Napisati program koji za učitani karakter ispisuje uneti karakter i njegov ASCII kod. Ukoliko je uneti karakter malo (veliko) slovo, ispisati i odgovarajuće veliko (malo) slovo i njegov ASCII kod.

Primer 1

```
Interakcija sa programom:
 Unesite karakter: 0
 Uneti karakter: 0
 ASCII kod: 48
```

Primer 2

```
Interakcija sa programom:
 Unesite karakter: ?
 Uneti karakter: ?
 ASCII kod: 63
```

Primer 3

```
Interakcija sa programom:
 Unesite karakter: A
 Uneti karakter: A
 ASCII kod: 65
 Odgovarajuce malo slovo: a
 ASCII kod: 97
```

Primer 4

```
Interakcija sa programom:
 Unesite karakter: v
 Uneti karakter: v
 ASCII kod: 118
 Odgovarajuce veliko slovo: V
 ASCII kod: 86
```

Zadatak 1.3.9 Napisati program koji učitava tri karaktera. Ispitati da li među unetim karakterima ima cifara i ako je tako odrediti proizvod tih cifara. Ukoliko među unetim karakterima nema cifara, program treba da ispiše odgovarajuću poruku. Napomena: *Karakteri koji se unose su razdvojeni blanko znacima.*

Primer 1

```
Interakcija sa programom:
 Unesite karaktere: A 5 3
 Proizvod cifara: 15
```

Primer 2

```
Interakcija sa programom:
 Unesite karaktere: k ! m
 Medju unetim karakterima nema cifara.
```

Primer 3

```
Interakcija sa programom:
 Unesite karaktere: 9 9 9
 Proizvod cifara: 729
```

Primer 4

```
Interakcija sa programom:
 Unesite karaktere: a 8 0
 Proizvod cifara: 0
```

Zadatak 1.3.10 Kasirka unosi šifru artikla koja se zadaje u formi tri spojena karaktera koji mogu biti mala slova, velika slova ili cifre. U kasi su sve šifre zapisane malim slovima i ciframa. Napisati program koji kasirkin unos konvertuje u unos koji je odgovarajući za kasu, tj. koji sva velika slova pretvara u odgovarajuća mala, a ostale karaktere ne menja. U slučaju neispravnog unosa šifre, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite sifru: aBc
 Rezultat: abc
```

Primer 2

```
Interakcija sa programom:
 Unesite sifru: a?!
 Greška: ? je neispravan karakter.
```

Primer 3

```
Interakcija sa programom:
 Unesite karaktere: 5A5
 Rezultat: 5a5
```

Primer 4

```
Interakcija sa programom:
 Unesite karaktere: 123
 Rezultat: 123
```

Zadatak 1.3.11 Napisati program koji za uneti četvorocifreni broj ispisuje njegovu najveću cifru. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 6835
 Najveca cifra je: 8
```

Primer 2

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 7777
 Najveca cifra je: 7
```

Primer 3

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 238
 Greska: niste uneli cetvorocifreni broj.
```

Primer 4

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: -2002
 Najveca cifra je: 2
```

Zadatak 1.3.12 Trocifreni broj je Armstrongov ako je jednak zbiru kubova svojih cifara. Napisati program koji za dati pozitivan trocifreni broj proverava da li je Armstrongov. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite pozitivan trocifreni broj:
 153
 Broj je Armstrongov.
```

Primer 2

```
Interakcija sa programom:
 Unesite pozitivan trocifreni broj:
 111
 Broj nije Armstrongov.
```

Primer 3

```
Interakcija sa programom:
 Unesite pozitivan trocifreni broj:
 84
 Greska: niste uneli pozitivan trocifreni broj.
```

Primer 4

```
Interakcija sa programom:
 Unesite pozitivan trocifreni broj:
 371
 Broj je Armstrongov.
```

Zadatak 1.3.13 Napisati program koji ispisuje proizvod parnih cifara unetog četvorocifrenog broja. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 8123
 Proizvod parnih cifara: 16
```

Primer 2

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 3579
 Nema parnih cifara.
```

Primer 3

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 288
 Greska: niste uneli cetvorocifreni broj.
```

Primer 4

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: -1234
 Proizvod parnih cifara: 8
```

Zadatak 1.3.14 Napisati program koji učitava četvorocifreni broj i ispisuje broj koji se dobija kada se unetom broju razmene najmanja i najveća cifra. U slučaju da se najmanja ili najveća cifra pojavljuju na više pozicija, uzeti prvo pojavljivanje, gledajući sa desna na levo. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 2863
 Rezultat: 8263
```

Primer 2

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 1192
 Rezultat: 1912
```

Primer 3

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 247
 Greska: niste uneli cetvorocifreni broj.
```

Primer 4

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: -4239
 Rezultat: -4932
```

Zadatak 1.3.15 Napisati program koji za uneti četvorocifreni broj proverava da li su njegove cifre uređene neopadajuće, nerastuće ili nisu uređene i štampa odgovarajuću poruku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 1389
 Cifre su uredjene neopadajuce.
```

Primer 2

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: -9622
 Cifre su uredjene nerastuce.
```

Primer 3

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 88
 Greska: niste uneli cetvorocifreni broj.
```

Primer 4

```
Interakcija sa programom:
 Unesite cetvorocifreni broj: 6792
 Cifre nisu uredjene.
```

Zadatak 1.3.16 Napisati program koji ispituje da li se tačke $A(x_1, y_1)$ i

$B(x_2, y_2)$ nalaze u istom kvadrantu. Koordinate tačaka su realni brojevi jednostruke tačnosti.

Primer 1

```
Interakcija sa programom:
 Unesite koordinate tacke A: 1.5 6
 Unesite koordinate tacke B: 2.33 9.8
 Tacke se nalaze u istom kvadrantu.
```

Primer 2

```
Interakcija sa programom:
 Unesite koordinate tacke A: -3 6
 Unesite koordinate tacke B: 0.33 -5
 Tacke se ne nalaze u istom kvadrantu.
```

Primer 3

```
Interakcija sa programom:
 Unesite koordinate tacke A: 0 -6
 Unesite koordinate tacke B: -1 -99.66
 Tacke se nalaze u istom kvadrantu.
```

Primer 4

```
Interakcija sa programom:
 Unesite koordinate tacke A: 3 -6
 Unesite koordinate tacke B: -0.33 0
 Tacke se ne nalaze u istom kvadrantu.
```

Zadatak 1.3.17 Napisati program koji ispituje da li se tačke $A(x_1, y_1)$, $B(x_2, y_2)$ i $C(x_3, y_3)$ nalaze na istoj pravoj.

Primer 1

```
Interakcija sa programom:
 Unesite koordinate tacke A: 1.5 6
 Unesite koordinate tacke B: -2.5 -10
 Unesite koordinate tacke C: 3 12
 Tacke se nalaze na istoj pravoj.
```

Primer 2

```
Interakcija sa programom:
 Unesite koordinate tacke A: -1.5 3
 Unesite koordinate tacke B: -0.4 9.8
 Unesite koordinate tacke C: 2 3
 Tacke se ne nalaze na istoj pravoj.
```

Primer 3

```
Interakcija sa programom:
 Unesite koordinate tacke A: 1.55 6
 Unesite koordinate tacke B: -8.4 9.8
 Unesite koordinate tacke C: 5 4.682412
 Tacke se nalaze na istoj pravoj.
```

Primer 4

```
Interakcija sa programom:
 Unesite koordinate tacke A: 5.5 3.5
 Unesite koordinate tacke B: 5.5 3.5
 Unesite koordinate tacke C: 5.5 3.5
 Tacke se nalaze na istoj pravoj.
```

Primer 5

```
Interakcija sa programom:
 Unesite koordinate tacke A: 1 2
 Unesite koordinate tacke B: 1 2
 Unesite koordinate tacke C: -56 1.3
 Tacke se nalaze na istoj pravoj.
```

Primer 6

```
Interakcija sa programom:
 Unesite koordinate tacke A: 3.4 3.5
 Unesite koordinate tacke B: -10 -1
 Unesite koordinate tacke C: -10 -1
 Tacke se nalaze na istoj pravoj.
```

Zadatak 1.3.18 Napisati program za rad sa intervalima. Za dva celobrojna intervala $[a_1, b_1]$ i $[a_2, b_2]$ program treba da odredi:

(a) dužinu preseka datih intervala

(b) presečni interval datih intervala

(c) dužinu dela prave koju pokrivaju dati intervali

(d) najmanji interval koji sadrži date intervale.

Primer 1

```
Interakcija sa programom:
 Unesite a1, b1, a2 i b2: 2 9 4 11
 Duzina preseka: 5
 Presecni interval: [4,9]
 Duzina koju pokrivaju: 9
 Najmanji interval: [2, 11]
```

Primer 2

```
Interakcija sa programom:
 Unesite a1, b1, a2 i b2: 1 2 10 13
 Duzina preseka: 0
 Presecni interval: prazan
 Duzina koju pokrivaju: 4
 Najmanji interval: [1, 13]
```

Zadatak 1.3.19 Napisati program koji za unete koeficijente kvadratne jednačine ispisuje koliko realnih rešenja jednačina ima i ako ih ima, ispisuje ih zaokružene na dve decimale.

Primer 1

```
Interakcija sa programom:
 Unesite koeficijente A, B i C: 1 3 2
 Jednacina ima dva razlicita realna resenja:
 -1.00 i -2.00
```

Primer 2

```
Interakcija sa programom:
 Unesite koeficijente A, B i C: 1 1 1
 Jednacina nema resenja.
```

Zadatak 1.3.20 U nizu 12345678910111213....9899 ispisani su redom brojevi od 1 do 99. Napisati program koji za uneti ceo broj k ($1 \leq k \leq 189$) ispisuje cifru koja se nalazi na k-toj poziciji datog niza. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite k: 13
 Na 13-toj poziciji je broj 1.
```

Primer 2

```
Interakcija sa programom:
 Unesite k: 105
 Na 105-toj poziciji je broj 7.
```

Primer 3

```
Interakcija sa programom:
 Unesite k: 200
 Greska: neispravan unos pozicije.
```

Primer 4

```
Interakcija sa programom:
 Unesite k: 10
 Na 10-toj poziciji je broj 1.
```

Zadatak 1.3.21 Data je funkcija $f(x) = 2 \cdot cos(x) - x^3$. Napisati program koji za učitanu vrednost realne promenljive x i vrednost celobrojne promenljive k koja može biti 1, 2 ili 3 izračunava vrednost funkcije $F(x, k)$ koja se dobija tako što se funkcija f primeni k-puta ($F(x,1) = f(x), F(x,2) = f(f(x)), F(x,3) = f(f(f(x)))$). Dobijenu vrednosti ispisati zaokruženu na dve decimale. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite redom x i k:
 2.31 2
 F(2.31, 2) = 2557.52
```

Primer 2

```
Interakcija sa programom:
 Unesite redom x i k:
 2.31 0
 Greska: nedozvoljena
 vrednost za k.
```

Primer 3

```
Interakcija sa programom:
 Unesite redom x i k:
 12 1
 F(12, 1) = -1726.31
```

Zadatak 1.3.22 Napisati program koji za uneti redni broj dana u nedelji ispisuje ime odgovarajućeg dana. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 4
 U pitanju je: cetvrtak
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 8
 Greska: neispravan unos
 dana.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 7
 U pitanju je: nedelja
```

Zadatak 1.3.23 Napisati program koji za uneti karakter ispituje da li je samoglasnik ili ne.

Primer 1

```
Interakcija sa programom:
 Unesite jedan karakter: A
 Uneti karakter je samoglasnik.
```

Primer 2

```
Interakcija sa programom:
 Unesite jedan karakter: i
 Uneti karakter je samoglasnik.
```

Primer 3

```
Interakcija sa programom:
 Unesite jedan karakter: f
 Uneti karakter nije samoglasnik.
```

Primer 4

```
Interakcija sa programom:
 Unesite jedan karakter: 4
 Uneti karakter nije samoglasnik.
```

Zadatak 1.3.24 Napisatiti program koji učitava dva cela broja i jedan od karaktera +, -, *, / ili % i ispisuje vrednost izraza dobijenog primenom date operacije nad učitanim vrednostima. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite izraz: 8 - 11
 Rezultat je: -3
```

Primer 2

```
Interakcija sa programom:
 Unesite izraz: 14 / 0
 Greska: deljenje nulom.
```

Primer 3

```
Interakcija sa programom:
 Unesite izraz: 5 ? 7
 Greska: nepoznat operator.
```

Primer 4

```
Interakcija sa programom:
 Unesite izraz: 19 / 5
 Rezultat je: 3
```

Zadatak 1.3.25 Napisati program koji za uneti datum u formatu *dan.mesec.* ispisuje godišnje doba kojem pripadaju. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite dan i mesec: 14.10.
 jesen
```

Primer 2

```
Interakcija sa programom:
 Unesite dan i mesec: 2.8.
 leto
```

Primer 3

```
Interakcija sa programom:
 Unesite dan i mesec: 27.2.
 zima
```

Zadatak 1.3.26 Napisati program koji za unetu godinu i mesec ispisuje naziv meseca kao i koliko dana ima u tom mesecu te godine. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite godinu: 2018
 Unesite mesec: 1
 Januar, 31 dan
```

Primer 2

```
Interakcija sa programom:
 Unesite godinu: 2000
 Unesite mesec: 2
 Februar, 29 dana
```

Primer 3

```
Interakcija sa programom:
 Unesite godinu: 2018
 Unesite mesec: 13
 Greska: neispravan unos
 meseca.
```

Zadatak 1.3.27 Napisati program koji za uneti datum u formatu *dan.mesec.godina.* proverava da li je korektan.

Primer 1

```
Interakcija sa programom:
 Unesite datum: 25.11.1983.
 Datum je korektan.
```

Primer 2

```
Interakcija sa programom:
 Unesite datum: 1.17.2004.
 Datum nije korektan.
```

Zadatak 1.3.28 Napisati program koji za korektno unet datum u formatu *dan.mesec.godina.* ispisuje datum prethodnog dana.

Primer 1

```
Interakcija sa programom:
 Unesite datum:
 30.4.2008.
 Prethodni datum:
 29.4.2008.
```

Primer 2

```
Interakcija sa programom:
 Unesite datum:
 1.12.2005.
 Prethodni datum:
 30.11.2005.
```

Primer 3

```
Interakcija sa programom:
 Unesite datum:
 1.1.2019.
 Prethodni datum:
 31.12.2018.
```

Zadatak 1.3.29 Napisati program koji za korektno unet datum u formatu *dan.mesec.godina.* ispisuje datum narednog dana.

Primer 1

```
Interakcija sa programom:
 Unesite datum:
 30.4.2008.
 Naredni datum:
 1.5.2008.
```

Primer 2

```
Interakcija sa programom:
 Unesite datum:
 1.12.2005.
 Naredni datum:
 2.12.2005.
```

Primer 3

```
Interakcija sa programom:
 Unesite datum:
 31.12.2008.
 Naredni datum:
 1.1.2009.
```

*** Zadatak 1.3.30** Polje šahovske table se definiše parom celih brojeva (x, y), $1 \leq x, y \leq 8$, gde je x redni broj reda, a y redni broj kolone. Napisati program koji za unete parove (k, l) i (m, n) proverava

(a) da li su polja (k, l) i (m, n) iste boje

(b) da li kraljica sa (k, l) ugrožava polje (m, n)

(c) da li konj sa (k, l) ugrožava polje (m, n)

Pretpostaviti da je polje $(1, 1)$ crno i da predstavlja donji levi ugao šahovske table. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite (k,l): 1 1
 Unesite (m,n): 2 2
 Polja su iste boje.
 Kraljica sa (1,1) ugrozava (2,2).
 Konj sa (1,1) ne ugrozava (2,2).
```

Primer 2

```
Interakcija sa programom:
 Unesite (k,l): 1 1
 Unesite (m,n): 3 2
 Polja su razlicite boje.
 Kraljica sa (1,1) ne ugrozava (3,2).
 Konj sa (1,1) ugrozava (3,2).
```

Primer 3

```
Interakcija sa programom:
 Unesite (k,l): 5 4
 Unesite (m,n): 3 3
 Polja su razlicite boje.
 Kraljica sa (5,4) ne ugrozava (3,3).
 Konj sa (5,4) ugrozava (3,3).
```

Primer 4

```
Interakcija sa programom:
 Unesite (k,l): 0 1
 Unesite (m,n): 3 9
 Greska: neispravna pozicija.
```

1.4 Rešenja

Rešenje 1.3.1

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a, b, c, najmanji;

  /* Ucitavanje ulaznih vrednosti. */
  printf("Unesite tri cela broja: ");
  scanf("%d%d%d", &a, &b, &c);

  /* Inicijalizovanje najmanjeg broja na vrednost prvog broja. */
  najmanji = a;

  /* Azuriranje vrednosti minimuma u slucaju da je vrednost drugog
     broja manja od vrednosti tekuceg minimuma. */
  if (b < najmanji)
    najmanji = b;

  /* Ponavljanje postupka za treci broj. */
  if (c < najmanji)
    najmanji = c;

  /* Ispis rezultata. */
  printf("Najmanji: %d\n", najmanji);

  return 0;
}
```

Rešenje 1.3.2

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float x, apsolutno_x;

  /* Ucitavanje vrednosti broja. */
  printf("Unesite jedan realan broj:");
  scanf("%f", &x);

  /* Racunanje apsolutne vrednosti unetog broja. */
  apsolutno_x = x;
  if (x < 0)
    apsolutno_x = -x;

  /* Ispis rezultata. */
```

```
  printf("Apsolutna vrednost: %.2f\n", apsolutno_x);

  /* II nacin: koriscenjem funkcije fabs cija se deklaracija nalazi
     u zaglavlju math.h: apsolutno_x=fabs(x); */

  return 0;
}
```

Rešenje 1.3.3

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x;
  float reciprocno_x;

  /* Ucitavanje vrednosti broja x. */
  printf("Unesite jedan ceo broj:");
  scanf("%d", &x);

  /* Provera ispravnosti ulaznih podataka. Napomena: Za
     razliku od izlaza iz programa sa kodom 0 (return 0;) koji
     sluzi kao indikator da se program zavrsio uspesno, izlaz iz
     programa sa izlaznim kodom razlicitim od nule sluzi kao
     indikator da je pri izvrsavanju programa doslo do neke greske.
    */
  if (x == 0) {
    printf("Greska: nedozvoljeno je deljenje nulom.\n");
    return 1;
  }

  /* Racunanje reciprocne vrednosti. */
  reciprocno_x = 1.0 / x;

  /* Ispis rezultata. */
  printf("Reciprocna vrednost: %.4f\n", reciprocno_x);

  return 0;
}
```

Rešenje 1.3.4

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a, b, c, suma;

  /* Ucitavanje ulaznih vrednosti. */
```

```
  printf("Unesite tri cela broja:");
  scanf("%d%d%d", &a, &b, &c);

  /* Inicijalizovanje sume na nulu. */
  suma = 0;

  /* Na sumu se dodaju vrednosti onih brojeva cija je vrednost
     pozitivna. Uvecavanje je moguce uraditi na dva nacina:
     I nacin: suma = suma + vrednost;
     II nacin: suma += vrednost; */
  if (a > 0)
    suma = suma + a;

  if (b > 0)
    suma += b;

  if (c > 0)
    suma += c;

  /* Ispis rezultata. */
  printf("Zbir pozitivnih: %d\n", suma);

  return 0;
}
```

Rešenje 1.3.5

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int cena1, cena2, cena3, najjeftiniji;
  int cena_bez_popusta, cena_sa_popustom;

  /* Ucitavanje vrednosti cena. */
  printf("Unesite tri cene: ");
  scanf("%d%d%d", &cena1, &cena2, &cena3);

  /* Provera ispravnosti ulaznih podataka. */
  if (cena1 <= 0 || cena2 <= 0 || cena3 <= 0) {
    printf("Greska: neispravan unos cene.");
    return 1;
  }

  /* Racunanje vrednosti najjeftinijeg artikla. */
  najjeftiniji = cena1;

  if (cena2 < najjeftiniji)
    najjeftiniji = cena2;

  if (cena3 < najjeftiniji)
```

```
25      najjeftiniji = cena3;

27    /* Racunanje cene sa i bez popusta. */
      cena_bez_popusta = cena1 + cena2 + cena3;
29    cena_sa_popustom = cena_bez_popusta - najjeftiniji + 1;

31    /* Ispis rezultata. */
      printf("Cena sa popustom: %d din\n", cena_sa_popustom);
33    printf("Usteda: %d din\n", cena_bez_popusta - cena_sa_popustom);

35    return 0;
  }
```

Rešenje 1.3.6

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracije potrebnih promenljivih. */
5    int sati, minuti;
     int preostali_sati, preostali_minuti;
7
     /* Ucitavanje podataka o vremenu. Napomena: Vreme se zadaje u
9       formatu sat:minut. Iz tog razloga je i odgovarajuci format u
        funkciji scanf %d:%d. */
11   printf("Unesite vreme: ");
     scanf("%d:%d", &sati, &minuti);
13
     /* Provera ispravnosti ulaznih podataka. */
15   if (sati > 24 || sati < 0 || minuti > 59 || minuti < 0) {
       printf("Greska: neispravan unos vremena.\n");
17     return 1;
     }
19
     /* Racunanje preostalog vremena. */
21   preostali_sati = 24 - sati - 1;
     preostali_minuti = 60 - minuti;
23
     if (preostali_minuti == 60) {
25     /* Uvecavanje vrednosti broja za 1 se moze uraditi na vise
          nacina. Neki od njih su:
27        broj = broj + 1;
          broj += 1;
29        broj++; */
       preostali_sati++;
31     preostali_minuti = 0;
     }
33
     /* Ispis rezultata. */
35   printf("Do ponoci: %d sati i %d minuta\n",
            preostali_sati, preostali_minuti);
```

```
37
  return 0;
39 }
```

Rešenje 1.3.7

```
1 #include <stdio.h>

3 int main() {
  /* Deklaracija potrebne promenljive. */
5   unsigned int godina;

7   /* Ucitavanje vrednosti godine. */
  printf("Unesite godinu:");
9   scanf("%u", &godina);

11  /* Provera da li je godina prestupna i ispis odgovarajuce poruke.
      Godina je prestupna ukoliko vazi jedan od narednih uslova:
13      1. da je deljiva sa 4, a nije sa 100
      2. da je deljiva sa 400. */
15  if ((godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0)
    printf("Godina je prestupna.\n");
17  else
    printf("Godina nije prestupna.\n");
19
  return 0;
21 }
```

Rešenje 1.3.8

```
  #include <stdio.h>
2
  int main() {
4   /* Deklaracija potrebne promenljive. */
    char c;
6
    /* Ucitavanje jednog karaktera. */
8   printf("Unesite karakter: ");
    scanf("%c", &c);
10
    /* Ispis karaktera i vrednosti njegovog ASCII koda. */
12  printf("Uneti karakter: %c\n", c);
    printf("ASCII kod: %d\n", c);
14
    /* Karakteri koji odgovaraju velikim slovima su u ASCII tablici
16     smesteni sekvencijalno. Na primer, ASCII kod karaktera 'A' je
      65, 'B' je 66, ..., 'Z' je 90. Isto vazi i za mala slova: 'a'
18     je 97, 'b' je 98, ..., 'z' je 122.

20     Odavde, ako se vrsi provera da li je neki karakter veliko
```

```
     slovo, dovoljno je proveriti da li se njegov ASCII kod nalazi
22   izmedju ASCII kodova slova 'A' i slova 'Z'.

24   Dodatno, moze se primetiti da je razlika izmedju ASCII koda
     svakog malog i odgovarajuceg velikog slova konstanta koja ima
26   vrednost 'a'-'A', sto je isto sto i 'b'-'B', itd. Zbog toga,
     ako je potrebno od velikog slova dobiti malo, onda je
28   dovoljno ASCII kodu velikog slova dodati pomenutu konstantu.
     Za mala slova, vazi obrnuto - da bi se dobilo veliko slovo,
30   ova konstanta se oduzima. */

32  if (c >= 'A' && c <= 'Z') {
      printf("Odgovarajuce malo slovo: %c\n", c + ('a' - 'A'));
34    printf("ASCII kod: %d\n", c + ('a' - 'A'));
    }
36
    if (c >= 'a' && c <= 'z') {
38    printf("Odgovarajuce veliko slovo: %c\n", c - ('a' - 'A'));
      printf("ASCII kod: %d\n", c - ('a' - 'A'));
40  }

42  return 0;
}
```

Rešenje 1.3.9

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracija potrebnih promenljivih. */
5    unsigned int broj_cifara = 0;
     unsigned int proizvod_cifara = 1;
7    int c1, c2, c3;

9    /* I nacin ucitavanja ulaza: koriscenjem funkcije getchar()
        Funkcija getchar cita jedan karakter sa ulaza i vraca njegov
11      ASCII kod. Napomena: Razmaci su takodje karakteri i nece
        automatski biti preskoceni. Iz tog razloga se getchar poziva 5
13      puta u ovom primeru. Posto je poznato da su drugi i cetvrti
        karakter blanko znaci, nema potrebe da se cuva povratna
15      vrednost tih poziva. */
     printf("Unesite karaktere: ");
17   c1 = getchar();
     getchar();
19   c2 = getchar();
     getchar();
21   c3 = getchar();

23   /* II nacin ucitavanja ulaza: koriscenjem funkcije scanf()
        Blanko znaci se navode kao deo ocekivanog formata ulaza.
25      char c1, c2, c3;
```

```
     scanf("%c %c %c", &c1, &c2, &c3); */

  /* Pogresan nacin ucitavanja ulaza:
     scanf("%c%c%c", &c1, &c2, &c3);
     U ovom slucaju ce u c1 biti upisan prvi karakter, u c2
     blanko i u c3 drugi karakter. */

  /* Karakteri koji predstavljaju cifre su u ASCII tablici takodje
     smesteni sekvencijalno. Na primer, '0' ima ASCII kod 48, '1'
     49, ..., '9' ima ASCII kod 57.

     Odavde, ako se proverava da li je karakter cifra, dovoljno je
     proveriti da li se njegov ASCII kod nalazi izmedju '0' i '9'.

     Dodatno, ako je potrebno izracunati dekadnu vrednost karaktera
     koji je cifra, dovoljno je od ASCII koda tog karaktera,
     oduzeti ASCII kod karaktera '0'. Na primer, '4'-'0' = 52 - 48
     = 4. */

  /* Racunanje proizvoda onih karaktera koji su cifre. */
  if (c1 >= '0' && c1 <= '9') {
    proizvod_cifara *= (c1 - '0');
    broj_cifara++;
  }

  if (c2 >= '0' && c2 <= '9') {
    proizvod_cifara *= (c2 - '0');
    broj_cifara++;
  }

  if (c3 >= '0' && c3 <= '9') {
    proizvod_cifara *= (c3 - '0');
    broj_cifara++;
  }

  /* Ispis rezultata. */
  if (broj_cifara == 0)
    printf("Medju unetim karakterima nema cifara.\n");
  else
    printf("Proizvod cifara: %u\n", proizvod_cifara);

  return 0;
}
```

Rešenje 1.3.10

```
#include <stdio.h>
#include <ctype.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
```

```
  int c1, c2, c3;

  /* Ucitavanje sifre artikla. */
  printf("Unesite sifru: ");
  c1 = getchar();
  c2 = getchar();
  c3 = getchar();

  /* Funkcije islower, isupper i isdigit proveravaju da li je
     prosledjeni karakter malo slovo, veliko slovo ili cifra.
     Deklaracije ovih funkcija se nalaze u zaglavlju ctype.h.

     Ukoliko prvi karakter nije ni malo slovo ni veliko slovo, ni
     cifra, ispisuje se odgovarajuca poruka o gresci i izlazi se
     iz programa. */
  if (!islower(c1) && !isupper(c1) && !isdigit(c1)) {
    printf("Greska: %c je neispravan karakter.\n", c1);
    return 1;
  }

  /* Postupak se ponavlja za druga dva karaktera. */
  if (!islower(c2) && !isupper(c2) && !isdigit(c2)) {
    printf("Greska: %c je neispravan karakter.\n", c2);
    return 1;
  }

  if (!islower(c3) && !isupper(c3) && !isdigit(c3)) {
    printf("Greska: %c je neispravan karakter.\n", c3);
    return 1;
  }

  /* Funkcija tolower(c) radi sledece: ako je c veliko slovo, kao
     povratnu vrednost vraca odgovarajuce malo slovo, u suprotnom
     vraca c. Dakle, tolower('A') je 'a', a tolower('6') = '6',...

     Slicno, samo obrnuto, radi i funkcija toupper(c). Deklaracije
     ovih funkcija se, takodje, nalaze u zaglavlju ctype.h. */
  c1 = tolower(c1);
  c2 = tolower(c2);
  c3 = tolower(c3);

  /* Ispis rezultata. */
  printf("Rezultat: %c%c%c\n", c1, c2, c3);

  return 0;
}
```

Rešenje 1.3.11

```
#include <stdio.h>
#include <stdlib.h>
```

```
3
int main() {
5   /* Deklaracije potrebnih promenljivih. */
    int n;
7   char jedinica, desetica, stotina, hiljada, najveca_cifra;

9   /* Ucitavanje vrednosti broja n. */
    printf("Unesite cetvorocifreni broj: ");
11  scanf("%d", &n);

13  /* Da bi program radio ispravno i za negativne brojeve, koristi
      se apsolutna vrednost broja n. */
15  n = abs(n);

17  /* Provera ispravnosti ulaznih podataka. */
    if (n < 1000 || n > 9999) {
19    printf("Greska: niste uneli cetvorocifreni broj.\n");
      return 1;
21  }

23  /* Izdvajanje cifara broja n. */
    jedinica = n % 10;
25  desetica = (n / 10) % 10;
    stotina = (n / 100) % 10;
27  hiljada = n / 1000;

29  /* Racunanje najvece cifra broja n. */
    najveca_cifra = jedinica;
31
    if (desetica > najveca_cifra)
33    najveca_cifra = desetica;

35  if (stotina > najveca_cifra)
      najveca_cifra = stotina;
37
    if (hiljada > najveca_cifra)
39    najveca_cifra = hiljada;

41  /* Ispis rezultata. */
    printf("Najveca cifra je: %d\n", najveca_cifra);
43
    return 0;
45 }
```

Rešenje 1.3.12

```
#include <stdio.h>
2 #include <stdlib.h>

4 int main() {
    /* Deklaracije potrebnih promenljivih. */
```

```
  int n;
  char jedinica, desetica, stotina;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite pozitivan trocifreni broj: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaznih podataka. */
  if (n < 100 || n > 999) {
    printf("Greska: niste uneli pozitivan trocifreni broj.\n");
    return 1;
  }

  /* Izdvajanje cifara broja n. */
  jedinica = n % 10;
  desetica = (n / 10) % 10;
  stotina = n / 100;

  /* Ispis rezultata. */
  if (n == jedinica * jedinica * jedinica +
      desetica * desetica * desetica + stotina * stotina * stotina)
    printf("Broj je Armstrongov.\n");
  else
    printf("Broj nije Armstrongov.\n");

  return 0;
}
```

Rešenje 1.3.13

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, broj_parnih, proizvod_parnih;
  char jedinica, desetica, stotina, hiljada;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite cetvorocifreni broj: ");
  scanf("%d", &n);

  /* Da bi program radio ispravno i za negativne vrednosti,
     koristi se apsolutna vrednost broja n. */
  n = abs(n);

  /* Provera ispravnosti ulaznih podataka. */
  if (n < 1000 || n > 9999) {
    printf("Greska: niste uneli cetvorocifreni broj.\n");
    return 1;
  }
```

```
23   /* Izdvajanje cifara broja n. */
     jedinica = n % 10;
25   desetica = (n / 10) % 10;
     stotina = (n / 100) % 10;
27   hiljada = n / 1000;

29   /* Inicijalizacija brojaca i rezultata. */
     broj_parnih = 0;
31   proizvod_parnih = 1;

33   /* Za svaku cifru se vrsi provera da li je parna i ukoliko jeste
        tekuci rezultat se mnozi tekucom cifrom. */
35   if (jedinica % 2 == 0) {
       proizvod_parnih = proizvod_parnih * jedinica;
37     broj_parnih++;
     }
39
     if (desetica % 2 == 0) {
41     proizvod_parnih = proizvod_parnih * desetica;
       broj_parnih++;
43   }

45   if (stotina % 2 == 0) {
       proizvod_parnih = proizvod_parnih * stotina;
47     broj_parnih++;
     }
49
     if (hiljada % 2 == 0) {
51     proizvod_parnih = proizvod_parnih * hiljada;
       broj_parnih++;
53   }

55   /* Ispis rezultata. */
     if (broj_parnih == 0)
57     printf("Nema parnih cifara.\n");
     else
59     printf("Proizvod parnih cifara: %d\n", proizvod_parnih);

61   return 0;
   }
```

Rešenje 1.3.14

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   int main() {
5    /* Deklaracije potrebnih promenljivih. */
     int n, n_abs, rezultat;
7    char jedinica, desetica, stotina, hiljada;
```

```
  int najveca, najmanja, stepen_najvece, stepen_najmanje;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite cetvorocifreni broj: ");
  scanf("%d", &n);

  /* Da bi program radio ispravno i za negativne vrednosti,
     koristi se apsolutna vrednost broja n. */
  n_abs = abs(n);

  /* Provera ispravnosti ulaznih podataka. */
  if (n_abs < 1000 || n_abs > 9999) {
    printf("Greska: niste uneli cetvorocifreni broj.\n");
    return 1;
  }

  /* Izdvajanje cifara broja n. */
  jedinica = n_abs % 10;
  desetica = (n_abs / 10) % 10;
  stotina = (n_abs / 100) % 10;
  hiljada = n_abs / 1000;

  /* Po algoritmu za trazenje najvece/najmanje cifre (koji je
     prikazan u zadatku 2.1.11) pronalaze se najveca i najmanja
     cifra broja n, kao i pozicije na kojoj se one nalaze.
     Radi lakseg izracunavanja, pozicija se pamti kao stepen broja
     10. Na primer, pozicija cifre jedinica je 1, cifre desetica
     10, itd... */
  najveca = jedinica;
  stepen_najvece = 1;

  if (desetica > najveca) {
    najveca = desetica;
    stepen_najvece = 10;
  }

  if (stotina > najveca) {
    najveca = stotina;
    stepen_najvece = 100;
  }

  if (hiljada > najveca) {
    najveca = hiljada;
    stepen_najvece = 1000;
  }

  /* Racunanje najmanje cifre. */
  najmanja = jedinica;
  stepen_najmanje = 1;

  if (desetica < najmanja) {
    najmanja = desetica;
```

```
    stepen_najmanje = 10;
  }

  if (stotina < najmanja) {
    najmanja = stotina;
    stepen_najmanje = 100;
  }

  if (hiljada < najmanja) {
    najmanja = hiljada;
    stepen_najmanje = 1000;
  }

  /* Ideja: U broju 4179, najmanja cifra je 1 i njen stepen je 100,
     a najveca cifra je 9 i njen stepen je 1. Zamena mesta se vrsi
     tako sto se oduzme 9 i doda 1, a zatim oduzme 100 i doda 900. */
  rezultat = n_abs - najveca * stepen_najvece
    + najmanja * stepen_najvece - najmanja * stepen_najmanje
    + najveca * stepen_najmanje;

  /* Ako je pocetni broj bio negativan i rezultat treba da bude
     negativan. */
  if(n < 0)
    rezultat = -rezultat;

  /* Ispis rezultata. */
  printf("Rezultat: %d\n", rezultat);

  return 0;
}
```

Rešenje 1.3.15

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n;
  char jedinica, desetica, stotina, hiljada;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite cetvorocifreni broj: ");
  scanf("%d", &n);

  /* Da bi program radio ispravno i za negativne vrednosti,
     koristi se apsolutna vrednost broja n. */
  n = abs(n);

  /* Provera ispravnosti ulaznih podataka. */
  if (n < 1000 || n > 9999) {
```

```
    printf("Greska: niste uneli cetvorocifreni broj.\n");
    return 1;
  }

  /* Izdvajanje cifara broja n. */
  jedinica = n % 10;
  desetica = (n / 10) % 10;
  stotina = (n / 100) % 10;
  hiljada = n / 1000;

  /* Ispis rezultata. */
  if (hiljada <= stotina && stotina <= desetica &&
      desetica <= jedinica)
    printf("Cifre su uredjene neopadajuce. \n");
  else if (hiljada >= stotina && stotina >= desetica &&
           desetica >= jedinica)
    printf("Cifre su uredjene nerastuce. \n");
  else
    printf("Cifre nisu uredjene.\n");

  return 0;
}
```

Rešenje 1.3.16

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float xa, ya, xb, yb;

  /* Ucitavanje koordinata tacaka A i B. */
  printf("Unesite koordinate tacke A: ");
  scanf("%f%f", &xa, &ya);

  printf("Unesite koordinate tacke B: ");
  scanf("%f%f", &xb, &yb);

  /* Provera da li su obe tacke u istom kvadrantu i ispis
     odgovarajuce poruke. */
  if ((xa >= 0 && ya >= 0 && xb >= 0 && yb >= 0) ||
      (xa <= 0 && ya >= 0 && xb <= 0 && yb >= 0) ||
      (xa >= 0 && ya <= 0 && xb >= 0 && yb <= 0) ||
      (xa <= 0 && ya <= 0 && xb <= 0 && yb <= 0))
    printf("Tacke se nalaze u istom kvadrantu.\n");
  else
    printf("Tacke se ne nalaze u istom kvadrantu.\n");

  return 0;
}
```

Rešenje 1.3.17

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  float xa, ya, xb, yb, xc, yc;
  float k, n;

  /* Ucitavanje koordinata tacaka A, B i C. */
  printf("Unesite koordinate tacke A: ");
  scanf("%f%f", &xa, &ya);

  printf("Unesite koordinate tacke B: ");
  scanf("%f%f", &xb, &yb);

  printf("Unesite koordinate tacke C: ");
  scanf("%f%f", &xc, &yc);

  /* Ako su bilo koje dve tacke jednake, onda se sigurno sve tri
     nalaze na jednoj pravoj. */
  if ((xa == xb && ya == yb) ||
      (xa == xc && ya == yc) || (xb == xc && yb == yc)) {
    printf("Tacke se nalaze na istoj pravoj.\n");
    return 0;
  }

  /* Odredjivanje koeficijenta pravca k i odsecka na y osi n prave
     y = k*x + n koja prolazi kroz tacke A i B. Napomena: U
     slucaju kada je xb jednako xa, ova prava je paralelna sa y
     osom pa k ima vrednost beskonacno, a n vrednost 0, tj.
     jednacina prave je x = xa (sto je isto sto i x = xb). Da bi se
     izbeglo deljenje nulom (xb-xa), ovaj slucaj se posebno
     obradjuje. */
  if (xb != xa) {
    k = (yb - ya) / (xb - xa);
    n = ya - k * xa;
    /* Provera da li tacka C pripada pravoj y=k*x + n na
       kojoj se vec nalaze tacke A i B. */
    if (yc == k * xc + n)
      printf("Tacke se nalaze na istoj pravoj.\n");
    else
      printf("Tacke se ne nalaze na istoj pravoj.\n");
  } else {
    /* Provera da li se i tacka C nalazi na pravoj x = xb. */
    if (xc == xb)
      printf("Tacke se nalaze na istoj pravoj.\n");
    else
      printf("Tacke se ne nalaze na istoj pravoj.\n");
  }

  /* II nacin: Tacke su kolinearne ako je
```

```
51      |xa ya 1 |
        |xb yb 1 | = 0
53      |xc yc 1 |
        odnosno, ako je
55      xa*yb + ya*xc + xb*yc - ya*xb - xa*yc - yb*xc = 0

57      if(xa*yb + ya*xc + xb*yc - ya*xb - xa*yc - yb*xc == 0)
          printf("Tacke se nalaze na istoj pravoj. \n");
59      else
          printf("Tacke se ne nalaze na istoj pravoj. \n"); */
61
    return 0;
63  }
```

Rešenje 1.3.18

```
    #include <stdio.h>
2
    int main() {
4     /* Deklaracija potrebnih promenljivih. */
      int a1, a2, b1, b2;
6
      /* Ucitavanje granica intervala. */
8     printf("Unesite a1, b1, a2 i b2: ");
      scanf("%d%d%d%d", &a1, &b1, &a2, &b2);
10
      /* Racunanje i ispis trazenih vrednosti (u zavisnosti od
12       razlicitih polozaja dva intervala). */
      if (a1 <= a2 && b1 >= a2) {
14      /* I slucaj: intervali se seku i [a1,b1] je pre [a2,b2]. */
        printf("Duzina preseka:: %d\n", b1 - a2);
16      printf("Presecni interval: [%d, %d]\n", a2, b1);
        printf("Duzina koju pokrivaju: %d\n", b2 - a1);
18      printf("Najmanji interval: [%d, %d]\n", a1, b2);
      } else if (a2 <= a1 && b2 >= a1) {
20      /* II slucaj: intervali se seku i [a2,b2] je pre [a1,b1]. */
        printf("Duzina preseka:: %d\n", b2 - a1);
22      printf("Presecni interval: [%d, %d]\n", a1, b2);
        printf("Duzina koju pokrivaju: %d\n", b1 - a2);
24      printf("Najmanji interval: [%d, %d]\n", a2, b1);
      } else if (a1 >= a2 && b1 <= b2) {
26      /* III slucaj: interval [a1,b1] se nalazi unutar [a2,b2]. */
        printf("Duzina preseka:: %d\n", b1 - a1);
28      printf("Presecni interval: [%d, %d]\n", a1, b1);
        printf("Duzina koju pokrivaju: %d\n", b2 - a2);
30      printf("Najmanji interval: [%d, %d]\n", a2, b2);
      } else if (a2 >= a1 && b2 <= b1) {
32      /* IV slucaj: interval [a2,b2] se nalazi unutar [a1,b1]. */
        printf("Duzina preseka:: %d\n", b2 - a2);
34      printf("Presecni interval: [%d, %d]\n", a2, b2);
        printf("Duzina koju pokrivaju: %d\n", b1 - a1);
```

```
36      printf("Najmanji interval: [%d, %d]\n", a1, b1);
    } else {
38      /* V slucaj: intervali su disjunktni. */
      printf("Duzina preseka:: 0\n");
40      printf("Presecni interval: prazan\n");
      printf("Duzina koju pokrivaju: %d\n", b1 - a1 + b2 - a2);
42      if (a1 < a2)
          printf("Najmanji interval: [%d, %d]\n", a1, b2);
44      else
          printf("Najmanji interval: [%d, %d]\n", a2, b1);
46    }

48    return 0;
  }
```

Rešenje 1.3.19

```
1 #include <stdio.h>
  #include <math.h>
3
  int main() {
5   /* Deklaracija potrebnih promenljivih. */
    float a, b, c, D;
7
    /* Ucitavanje koeficijenata kvadratne jednacine. */
9   printf("Unesite koeficijente A, B i C:");
    scanf("%f%f%f", &a, &b, &c);
11
    /* Racunanje resenja jednacine u zavisnosti od vrednosti
13     koeficijenata a, b i c i ispis rezultata. */
    if (a == 0) {
15    if (b == 0) {
        if (c == 0) {
17        /* Slucaj a==0 && b==0 && c==0: beskonacno mnogo resenja. */
          printf("Jednacina ima beskonacno mnogo resenja\n");
19      } else {
          /* Slucaj a==0 && b==0 && c!=0: nema resenja. */
21        printf("Jednacina nema resenja\n");
        }
23    } else {
        /* Slucaj a=0 && b!=0: jedinstveno resenje. */
25      printf("Jednacina ima jedinstveno realno resenje %.2f\n",
              -c / b);
27    }
    } else {
29    /* Slucaj a != 0: Racunanje diskriminante. */
      D = b * b - 4 * a * c;
31
      /* U zavisnosti od vrednosti diskriminante, ispisuje se
33       rezultat. */
      if (D < 0) {
```

```
35        printf("Jednacina nema realnih resenja\n");
      } else if (D > 0) {
37        printf("Jednacina ima dva realna resenja %.2f i %.2f\n",
                 (-b + sqrt(D)) / (2 * a), (-b - sqrt(D)) / (2 * a));
39    } else {
        printf("Jednacina ima jedinstveno realno resenje %.2f\n",
41               -b / (2 * a));
      }
43  }

45  return 0;
}
```

Rešenje 1.3.20

```
1 #include <stdio.h>

3 int main() {
    /* Deklaracija potrebnih promenljivih. */
5   int k, broj;

7   /* Ucitavanje trazene pozicije. */
    printf("Unesite k: ");
9   scanf("%d", &k);

11  /* Provera ispravnosti ulaznih podataka. */
    if (k < 1 || k > 189) {
13    printf("Greska: neispravan unos pozicije.\n");
      return 1;
15  }

17  /* Racunanje rezultata. */
    if (k < 10) {
19    /* I slucaj: trazi se jednocifreni broj. */
      printf("Na %d-toj poziciji je broj %d.\n", k, k);
21  } else {
      /* II slucaj: trazi se dvocifreni broj. */
23
      /* Ideja: izracunati broj na koji pokazuje pozicija k. Zatim,
25       ako je k parno, uzeti cifru desetica tog broja, a ako je k
         neparno, uzeti cifru jedinica tog broja.
27
         Na primer, za k=14 i k=15, broj koji se nalazi na ovim
29       pozicijama je 12, pa u slucaju da je k=14, treba ispisati 1,
         a u slucaju da je k=15, treba ispisati 2. */
31
      /* Odredjivanje odgovarajuceg broja: Kada bi niz izgledao
33       10111213...9899, za dato k, broj bi se dobio kao 9 + k/2 + 1
         za neparne vrednosti k, odnosno 9 + k/2 za parne (dodaje se
35       vrednost deset jer je prvi broj u nizu desetka.) Na primer:
         k=1, broj = 9 + 1/2 + 1 = 9 + 0 + 1 = 10 k=2, broj = 9 + 2/2
```

```
37        = 10 k=3, broj = 9 + 3/2 + 1 = 9 + 1 + 1 = 11 k=4, broj = 9
          + 4/2 = 11 ... Posto ovde postoji i 9 pozicija ispred,
39        potrebno je i njih uzeti u obzir - odatle: broj = 9 +
          (k-9)/2 + 1 za neparne vrednosti k, odnosno broj = 9 +
41        (k-9)/2 za parne vrednosti k. */
      if (k % 2 != 0) {
43      broj = 9 + (k - 9) / 2;
        printf("Na %d-toj poziciji je broj %d.\n", k, broj % 10);
45    } else {
        broj = 9 + (k - 9) / 2 + 1;
47      printf("Na %d-toj poziciji je broj %d.\n", k, broj / 10);
      }
49  }

51  return 0;
}
```

Rešenje 1.3.21

```
#include <stdio.h>
2 #include <math.h>

4 int main() {
    /* Deklaracije potrebnih promenljivih. */
6   float x, Fx;
    int k;
8
    /* Ucitavanje vrednosti x i k. */
10  printf("Unesite redom x i k: ");
    scanf("%f %d", &x, &k);
12
    /* Provera ispravnosti ulaznih podataka. */
14  if (k < 1 || k > 3) {
      printf("Greska: nedozvoljena vrednost za k.\n");
16    return 0;
    }
18
    /* U zavisnosti od vrednosti k, data funkcija ce se izracunati
20     jednom, dva puta ili tri puta. */
    Fx = 2 * cos(x) - x * x * x;
22  if (k > 1)
      Fx = 2 * cos(Fx) - Fx * Fx * Fx;
24  if (k > 2)
      Fx = 2 * cos(Fx) - Fx * Fx * Fx;
26
    /* Ispis rezultata. Napomena: Ispis realnih brojeva sa %g
28     rezultuje ispisom na onaj broj decimala koliko sam broj ima.
       Dakle, broj 1 ce se ispisati kao 1, broj 2.33 kao 2.33, broj
30     0.9999 kao 0.9999. */
    printf("F(%g, %d) = %.2f\n", x, k, Fx);
32
```

```
  return 0;
}
```

Rešenje 1.3.22

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int dan;

  /* Ucitavanje rednog broja dana u nedelji. */
  printf("Unesite broj: ");
  scanf("%d", &dan);

  /*I nacin: Koriscenjem if-else naredbe.
  if(dan == 1)
    printf("ponedeljak\n");
  else if(dan == 2)
    printf("utorak\n");
  else if(dan == 3)
    printf("sreda\n");
  else if(dan == 4)
    printf("cetvrtak\n");
  else if(dan == 5)
    printf("petak\n");
  else if(dan == 6)
    printf("subota\n");
  else if(dan == 7)
    printf("nedelja\n");
  else
    printf("Greska: neispravan unos dana.\n"); */

  /* II nacin: Koriscenjem switch naredbe.*/
  switch (dan) {
  case 1:
    /* Ako dan ima vrednost 1, ispisuje se ponedeljak. */
    printf("ponedeljak\n");

    /* Ako se naredba break ne navede, izvrsice se i sledeca
       naredba, tj. ispis ce biti "ponedeljak utorak". */
    break;
  case 2:
    /* Postupak se ponavlja i za ostale dane. */
    printf("utorak\n");
    break;
  case 3:
    printf("sreda\n");
    break;
  case 4:
    printf("cetvrtak\n");
```

```
47      break;
    case 5:
49      printf("petak\n");
        break;
51    case 6:
        printf("subota\n");
53      break;
    case 7:
55      printf("nedelja\n");
        break;
57    default:
        /* Ako vrednost promenljive dan nije ni jedna od vrednosti
59         izmedju 1 i 7, onda je uneta vrednost neispravna. */
      printf("Greska: neispravan unos dana.\n");
61    }

63    return 0;
  }
```

Rešenje 1.3.23

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracija potrebne promenljive. */
5    char c;

7    /* Ucitavanje jednog karaktera. */
     printf("Unesite jedan karakter:");
9    scanf("%c", &c);

11   /* Proverava se da li je karakter c samoglasnik, tj. da li
        odgovara nekom od sledecih karaktera: A,E,I,O,U,a,e,i,o,u. */
13   switch (c) {
     case 'A':
15   case 'E':
     case 'I':
17   case 'O':
     case 'U':
19   case 'a':
     case 'e':
21   case 'i':
     case 'o':
23   case 'u':
       printf("Uneti karakter je samoglasnik.\n");
25     break;
     default:
27     printf("Uneti karakter nije samoglasnik.\n");
       break;
29   }
```

```
31   return 0;
}
```

Rešenje 1.3.24

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char op;
  int x, y;

  /* Ucitavanje izraza. */
  printf("Unesite izraz: ");
  scanf("%d %c %d", &x, &op, &y);

  /* Racunanje vrednosti izraza u zavisnosti od unete operacije. */
  switch (op) {
  case '+':
    printf("Rezultat je: %d\n", x + y);
    break;
  case '-':
    printf("Rezultat je: %d\n", x - y);
    break;
  case '*':
    printf("Rezultat je: %d\n", x * y);
    break;
  case '/':
    if (y == 0)
      printf("Greska: deljenje nulom.\n");
    else
      printf("Rezultat je: %d\n", x / y);
    break;
  case '%':
    printf("Rezultat je: %d\n", x % y);
    break;
  default:
    printf("Greska: nepoznat operator.\n");
  }

  return 0;
}
```

Rešenje 1.3.25

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int dan, mesec;
```

```
7   /* Ucitavanje datuma koji je zadat u formatu: dan.mesec. */
    printf("Unesite dan i mesec");
9   scanf("%d.%d.", &dan, &mesec);

11  /* Racunanje godisnjeg doba. */
    switch (mesec) {
13  case 1:
    case 2:
15    /* Ako je mesec januar ili februar, onda je sigurno u pitanju
         zima. */
17    printf("zima\n");
      break;
19  case 3:
      /* Ako je mesec mart, onda se godisnje doba odredjuje u
21       zavisnosti od dana u mesecu. */
      if (dan < 21)
23      printf("zima\n");
      else
25      printf("prolece\n");
      break;
27  case 4:
    case 5:
29    /* Ako je mesec april ili maj, onda je sigurno u pitanju
         prolece. */
31    printf("prolece\n");
      break;
33  case 6:
      /* Ako je mesec jun, onda se godisnje doba odredjuje u
35       zavisnosti od dana u mesecu. */
      if (dan < 21)
37      printf("prolece\n");
      else
39      printf("leto\n");
      break;
41  case 7:
    case 8:
43    /* Ako je mesec jul ili avgust, onda je sigurno u pitanju
         leto. */
45    printf("leto\n");
      break;
47  case 9:
      /* Ako je mesec septembar, onda se godisnje doba odredjuje u
49       zavisnosti od dana u mesecu. */
      if (dan < 23)
51      printf("leto\n");
      else
53      printf("jesen\n");
      break;
55  case 10:
    case 11:
57    /* Ako je mesec oktobar ili novembar, onda je sigurno u pitanju
```

```
        jesen. */
59      printf("jesen\n");
        break;
61    case 12:
        /* Ako je mesec decembar, onda se godisnje doba odredjuje u
63         zavisnosti od dana u mesecu. */
        if (dan < 22)
65        printf("jesen\n");
        else
67        printf("zima\n");
    }
69
    return 0;
71  }
```

Rešenje 1.3.26

```
1   #include <stdio.h>

3   int main() {
      /* Deklaracija potrebnih promenljivih. */
5     int godina, mesec, prestupna;

7     /* Ucitavanje vrednosti godine. */
      printf("Unesite godinu: ");
9     scanf("%d", &godina);

11    /* Provera ispravnosti ulaznih podataka. */
      if (godina < 0) {
13      printf("Greska: neispravan unos godine.\n");
        return 1;
15    }

17    /* Provera da li je godina prestupna, zbog februara. */
      if ((godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0)
19      prestupna = 1;
      else
21      prestupna = 0;

23    /* Ucitavanje rednog broja meseca. */
      printf("Unesite redni broj meseca: ");
25    scanf("%d", &mesec);

27    /* Ispis rezultata u zavisnosti od vrednosti meseca. */
      switch (mesec) {
29    case 1:
        printf("Januar, 31 dan\n");
31      break;
      case 2:
33      if (prestupna)
          printf("Februar, 29 dana\n");
```

```
35      else
          printf("Februar, 28 dana\n");
37      break;
    case 3:
39      printf("Mart, 31 dan\n");
        break;
41  case 4:
        printf("April, 30 dana\n");
43      break;
    case 5:
45      printf("Maj, 31 dan\n");
        break;
47  case 6:
        printf("Jun, 30 dana\n");
49      break;
    case 7:
51      printf("Jul, 31 dan\n");
        break;
53  case 8:
        printf("Avgust, 31 dan\n");
55      break;
    case 9:
57      printf("Septembar, 30 dana\n");
        break;
59  case 10:
        printf("Oktobar, 31 dan\n");
61      break;
    case 11:
63      printf("Novembar, 30 dana\n");
        break;
65  case 12:
        printf("Decembar, 31 dan\n");
67      break;
    default:
69      printf("Greska: neispravan unos meseca.\n");
        return 1;
71  }

73  return 0;
}
```

Rešenje 1.3.27

```
1 #include <stdio.h>

3 int main() {
    /* Deklaracija potrebnih promenljivih. */
5   int dan, mesec, godina, dozvoljeni_broj_dana;

7   /* Ucitavanje datuma. */
    printf("Unesite datum: ");
```

```
9   scanf("%d.%d.%d", &dan, &mesec, &godina);

11  /* Provera korektnosti vrednosti unete godine. */
    if (godina < 0) {
13    printf("Datum nije korektan.\n");
      return 0;
15  }

17  /* Provera korektnosti vrednosti unetog meseca. */
    if (mesec < 1 || mesec > 12) {
19    printf("Datum nije korektan.\n");
      return 0;
21  }

23  /* Provera korektnosti vrednosti unetog dana. */
    switch (mesec) {
25  case 1:
    case 3:
27  case 5:
    case 7:
29  case 8:
    case 10:
31  case 12:
      /* Dozvoljeni broj dana za januar, mart, maj, jul, avgust,
33       oktobar i decembar je 31. */
      dozvoljeni_broj_dana = 31;
35    break;
    case 2:
37    /* Dozvoljeni broj dana za februar je 28 ili 29 u zavisnosti od
         toga da li je godina prestupna ili ne. */
39    if ((godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0)
        dozvoljeni_broj_dana = 29;
41    else
        dozvoljeni_broj_dana = 28;
43    break;
    case 4:
45  case 6:
    case 9:
47  case 11:
      /* Dozvoljeni broj dana za april, jun, septembar i novembar je
49       30. */
      dozvoljeni_broj_dana = 30;
51    break;
    }
53
    if (dan < 0 || dan > dozvoljeni_broj_dana) {
55    printf("Datum nije korektan.\n");
      return 0;
57  }

59  /* Kako su sve provere korektnosti prosle, datum se smatra
       korektnim. */
```

```
61   printf("Datum je korektan.\n");

63   return 0;
}
```

Rešenje 1.3.28

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracije potrebnih promenljivih. */
5    int dan, mesec, godina;
     int prethodni_dan, prethodni_mesec, prethodni_godina;
7
     /* Ucitavanje datuma. */
9    printf("Unesite datum: ");
     scanf("%d.%d.%d.", &dan, &mesec, &godina);
11
     /* Racunanje dana, meseca i godine prethodnog dana. */
13   prethodni_dan = dan - 1;
     prethodni_mesec = mesec;
15   prethodni_godina = godina;

17   /* Ako je potrebno, vrse se korekcije. */
     if (prethodni_dan == 0) {
19     prethodni_mesec = mesec - 1;
       if (prethodni_mesec == 0) {
21       prethodni_mesec = 12;
         prethodni_godina = godina - 1;
23     }

25     switch (prethodni_mesec) {
       case 1:
27     case 3:
       case 5:
29     case 7:
       case 8:
31     case 10:
       case 12:
33       prethodni_dan = 31;
         break;
35     case 2:
         if ((prethodni_godina % 4 == 0 && prethodni_godina % 100 != 0)
37           || prethodni_godina % 400 == 0)
           prethodni_dan = 29;
39       else
           prethodni_dan = 28;
41       break;
       case 4:
43     case 6:
       case 9:
```

```
45      case 11:
          prethodni_dan = 30;
47      }
    }
49
    /* Ispis rezultata. */
51  printf("Prethodni datum: %d.%d.%d.\n",
           prethodni_dan, prethodni_mesec, prethodni_godina);
53
    return 0;
55  }
```

Rešenje 1.3.29 Pogledajte zadatak 1.3.28.

Rešenje 1.3.30

```
#include <stdio.h>
2 #include<stdlib.h>

4 int main() {
    /* Deklaracija potrebnih promenljivih. */
6   int k, l, m, n;

8   /* Ucitavanje vrednosti pozicija na tabli. */
    printf("Unesite (k,l): ");
10  scanf("%d%d", &k, &l);

12  printf("Unesite (m,n): ");
    scanf("%d%d", &m, &n);
14
    /* Provera ispravnosti ulaznih podataka. */
16  if (k < 1 || k > 8 || l < 1 || l > 8 ||
        m < 1 || m > 8 || n < 1 || n > 8) {
18    printf("Greska: neispravna pozicija.\n");
      return 1;
20  }

22  if(k == m && l == n){
      printf("Greska: pozicije moraju biti razlicite.\n");
24    return 1;
    }
26
    /* Provera da li su (k,l) i (m,n) iste boje. Polja su iste
28     boje ako su:
       1) oba reda parna i obe kolone parne ILI
30     2) oba reda neparna i obe kolone neparne. */
    if (((k % 2 == m % 2) && (l % 2 == n % 2))
32      || ((k % 2 != m % 2) && (l % 2 != n % 2)))
      printf("Polja su iste boje.\n");
34  else
      printf("Polja su razlicite boje.\n");
```

```
36   /* II nacin:
        if((k+l) % 2 == (m+n) % 2)
38        printf("Polja su iste boje.\n");
        else
40        printf("Polja su razlicite boje.\n"); */

42   /* Provera da li kraljica sa (k,l) napada polje (m,n).
        Kraljica napada polje u sledecim situacijama:
44      1) Ako se nalaze u istom redu (k==m)
        2) Ako se nalaze u istoj koloni (l==n)
46      3) Ako se nalaze na istoj dijagonali. Dijagonala moze biti:
           a) paralelna glavnoj dijagonali (k-l == m-n)
48         b) paralelna sporednoj dijagonali (k+l == m+n) */
     if ((k == m) || (l == n) || (k - l == m - n)
50       || (k + l == m + n)) {
       printf("Kraljica sa (%d, %d) ugrozava (%d, %d).\n",
52             k, l, m, n);
     }
54   else {
       printf("Kraljica sa (%d, %d) ne ugrozava (%d, %d).\n",
56             k, l, m, n);
     }
58
     /* Provera da li konj sa (k, l) napada polje (m, n). Postoji
60      8 mogucih vrednosti za polja koja konj napada. Vrsi se
        provera da li je (m,n) jednako nekom od tih polja. */
62   if ((abs(k-m) == 2 &&  abs(n-l) == 1) || (abs(n-l) == 2 && abs(m-k)
        == 1))
       printf("Konj sa (%d, %d) ugrozava (%d, %d).\n",
64             k, l, m, n);
     else
66     printf("Konj sa (%d, %d) ne ugrozava (%d, %d).\n",
             k, l, m, n);
68
     return 0;
70 }
```

1.5 Petlje

Zadatak 1.5.1 Napisati program koji pet puta ispisuje tekst `Mi volimo da programiramo`.

Primer 1

```
Interakcija sa programom:
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
```

Zadatak 1.5.2 Napisati program koji učitava pozitivan ceo broj n i n puta ispisuje tekst `Mi volimo da programiramo`. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 6
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
 Mi volimo da programiramo.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: pogresan unos broja n.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -5
 Greska: pogresan unos broja n.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 1
 Mi volimo da programiramo.
```

Zadatak 1.5.3 Napisati program koji učitava nenegativan ceo broj n a potom ispisuje sve cele brojeve od 0 do n. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 4
 0 1 2 3 4
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: -10
 Greska: pogresan unos broja n.
```

Zadatak 1.5.4 Napisati program koji učitava dva cela broja n i m ($n \leq m$) i ispisuje sve cele brojeve iz intervala $[n, m]$. Pri rešavanju zadatka:

(a) koristiti `while` petlju

(b) koristiti `for` petlju

(c) koristiti `do-while` petlju

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite granice intervala: -2 4
 -2 -1 0 1 2 3 4
```

Primer 2

```
Interakcija sa programom:
 Unesite granice intervala: 10 6
 Greska: pogresan unos granica.
```

Zadatak 1.5.5 Napisati program koji učitava nenegativan ceo broj n i izračunava njegov faktorijel. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 18
 18! = 6402373705728000
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 8
 8! = 40320
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 40
 Pri racunanju 40! ce doci do prekoracenja.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: -5
 Greska: neispravan unos.
```

Zadatak 1.5.6 Napisati program koji učitava realan broj x i ceo nenegativan broj n i izračunava n-ti stepen broja x, tj. x^n. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 4 3
 Rezultat: 64.00000
```

Primer 2

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 5.8 5
 Rezultat: 6563.56768
```

Primer 3

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 11.43 -6
 Greska: neispravan unos broja n.
```

Primer 4

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 11.43 0
 Rezultat: 1.00000
```

Zadatak 1.5.7 Napisati program koji učitava realan broj x i ceo broj n i izračunava n-ti stepen broja x.

Primer 1

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 2 -3
 Rezultat: 0.125
```

Primer 2

```
Interakcija sa programom:
 Unesite redom brojeve x i n: -3 2
 Rezultat: 9.000
```

Zadatak 1.5.8 Pravi delioci celog broja su svi delioci osim jedinice i samog tog broja. Napisati program koji za uneti pozitivan ceo broj n ispisuje sve njegove prave delioce. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 100
 Pravi delioci: 2 4 5 10 20 25 50
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: -6
 Greska: neispravan unos.
```

Zadatak 1.5.9 Napisati program koji za uneti ceo broj ispisuje broj dobijen uklanjanjem svih nula sa desne strane unetog broja.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 12000
 Rezultat: 12
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 0
 Rezultat: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -1400
 Rezultat: -14
```

Primer 4

```
Interakcija sa programom:
 Unesite broj: 147
 Rezultat: 147
```

Zadatak 1.5.10 Napisati program koji učitava ceo broj i ispisuje njegove cifre u obrnutom poretku.

Primer 1

```
Interakcija sa programom:
 Unesite ceo broj: 6789
 Rezultat: 9 8 7 6
```

Primer 2

```
Interakcija sa programom:
 Unesite ceo broj: -892345
 Rezultat: 5 4 3 2 9 8
```

Zadatak 1.5.11 Napisati program koji za uneti pozitivan ceo broj ispisuje da li je on deljiv sumom svojih cifara. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 12
 Broj 12 je deljiv sa 3.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 2564
 Broj 2564 nije deljiv sa 17.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -4
 Greska: neispravan ulaz.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj: 0
 Greska: neispravan ulaz.
```

Zadatak 1.5.12 Knjigovođa vodi evidenciju o transakcijama jedne firme i treba da napiše izveštaj o godišnjem poslovanju te firme. Firma je tokom godine imala t transakcija. Transakcije su predstavljene celim brojevima i u slučaju da je vrednost transakcije pozitivna, ta transakcija označava prihod firme, a u slučaju da je negativna rashod. Napisati program koji učitava nenegativan ceo broj t i podatke o t transakcijama i zatim izračunava i ispisuje ukupan prihod, ukupan rashod i zaradu, odnosno gubitak, koji je firma ostvarila tokom godine. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj t: 7
 Unesite transakcije:
 8 -50 45 2007 -67 -123 14
 Prihod: 2074
 Rashod: -240
 Zarada: 1834
```

Primer 2

```
Interakcija sa programom:
 Unesite broj t: 5
 Unesite transakcije:
 -5 -20 -4 -200 -8
 Prihod: 0
 Rashod: -237
 Gubitak: 237
```

Primer 3

```
Interakcija sa programom:
 Unesite broj t: 2
 Unesite transakcije:
 120 -120
 Prihod: 120
 Rashod: -120
 Zarada: 0
```

Primer 4

```
Interakcija sa programom:
 Unesite broj t: -6
 Greska: neispravan unos.
```

Primer 5

```
Interakcija sa programom:
 Unesite broj n: 0
 Nema evidentiranih transakcija.
```

Zadatak 1.5.13 Napisati program koji učitava pozitivan ceo broj n, a potom i n celih brojeva. Izračunati i ispisati zbir onih brojeva koji su istovremeno neparni i negativni. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n brojeva:
 1 -5 -6 3 -11
 Zbir neparnih i negativnih: -16
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
 Unesite n brojeva:
 5 8 13 17
 Zbir neparnih i negativnih: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -4
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Zadatak 1.5.14 Napisati program koji učitava cele brojeve sve do unosa broja nula i ispisuje proizvod onih unetih brojeva koji su pozitivni.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve:
 -87 12 -108 -13 56 0
 Proizvod pozitivnih brojeva je 672.
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve:
 0
 Nije unet nijedan broj.
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve:
 -5 -200 -43 0
 Medju unetim brojevima nema pozitivnih.
```

Primer 4

```
Interakcija sa programom:
 Unesite brojeve:
 1 0
 Proizvod pozitivnih brojeva je 1.
```

Zadatak 1.5.15 Napisati program koji za uneti ceo broj proverava i ispisuje da li se cifra 5 nalazi u njegovom zapisu.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 1857
 Broj 1857 sadrzi cifru 5.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 84
 Broj 84 ne sadrzi cifru 5.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -2515
 Broj -2515 sadrzi cifru 5.
```

Zadatak 1.5.16 Napisati program koji učitava cele brojeve sve do unosa broja nula, a zatim izračunava i ispisuje aritmetičku sredinu unetih brojeva na četiri decimale.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve:
 8 5 6 3 0
 Aritmeticka sredina:
 5.5000
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve: 0
 Nisu uneti brojevi.
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve:
 762 -12 800 2010
 -356 899 -101 0
 Aritmeticka sredina:
 571.7143
```

Zadatak 1.5.17 U prodavnici se nalaze artikli čije su cene pozitivni realni brojevi. Napisati program koji učitava cene artikala sve do unosa broja nula i izračunava i ispisuje prosečnu vrednost cena u radnji. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite cene:
 8 5.2 6.11 3 0
 Prosecna cena: 5.5775
```

Primer 2

```
Interakcija sa programom:
 Unesite cene:
 6.32 -9
 Greska: neispravan unos
 cene.
```

Primer 3

```
Interakcija sa programom:
 Unesite cene:
 0
 Nisu unete cene.
```

Zadatak 1.5.18 Napisati program koji učitava pozitivan ceo broj n, a potom i n realnih brojeva. Program ispisuje koliko puta je prilikom unosa došlo do *promene znaka*. Do promena znaka dolazi kada se nakon pozitivnog broja unese negativan broj ili kada se nakon negativnog broja unese pozitivan broj. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 9
 Unesite brojeve:
 7.82 4.3 -1.2 56.8 -3.4 -72.1 8.9 11.2 -11.2
 Broj promena je 5.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite brojeve:
 -23.8 -11.2 0 5.6 7.2
 Broj promena je 1.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -6
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Zadatak 1.5.19 U prodavnici se nalazi n artikala čije su cene pozitivni realni brojevi. Napisati program koji učitava n, a potom i cenu svakog od n artikala i određuje i ispisuje najmanju cenu. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj artikla: 6
 Unesite cene artikala:
 12 3.4 90 100.53 53.2 12.8
 Najmanja cena: 3.400000
```

Primer 2

```
Interakcija sa programom:
 Unesite broj artikla: 3
 Unesite cene artikala:
 4 -8 92
 Greska: neispravan unos
 cene.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj artikla: -9
 Greska: neispravan unos.
```

Zadatak 1.5.20 Nikola želi da obraduje baku i da joj kupi jedan poklon u radnji. On na raspolaganju ima m dinara. U radnji se nalazi n artikala i zanima ga koliko ima artikala u radnji čija cena je manja ili jednaka m. Napisati program koji pomaže Nikoli da odredi broj takvih atikala. Program učitava realan nenegativan broj m, ceo nenegativan broj n i n pozitivnih realnih brojeva. Ispisati koliko artikala ima cenu čija je vrednost manja ili jednaka m. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Nikolin budzet: 12.37
 Unesite broj artikala: 5
 Unesite cene artikala: 11 54.13 6 13 8
 Ukupno artikala: 3
```

Primer 2

```
Interakcija sa programom:
 Nikolin budzet: 2
 Unesite broj artikala: 4
 Unesite cene artikala: 1 11 4.32 3
 Ukupno artikala: 1
```

Zadatak 1.5.21 Napisati program koji učitava ceo nenegativan broj n, n celih brojeva i zatim izračunava i ispisuje tražene vrednosti. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

(a) Broj sa najvećom cifrom desetica. Ukoliko ima više takvih, ispisati prvi.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite brojeve:
 18 365 25 1 78
 Broj sa najvecom cifrom desetica: 78.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 8
 Unesite brojeve:
 14 1576 -1267 -89 109 122 306 918
 Broj sa najvecom cifrom desetica: -89.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 4
 Unesite brojeve:
 100 200 300 400
 Broj sa najvecom cifrom desetica: 100.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: -12
 Greska: neispravan unos.
```

(b) Broj sa najvećim brojem cifara. Ukoliko ima više takvih, ispisati prvi.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n brojeva: 18 -365 251 1 78
 Najvise cifara ima broj -365.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 7
 Unesite n brojeva:
 3 892 18 21 639 742 85
 Najvise cifara ima broj 892.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 0
 Nisu uneti brojevi.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: -7
 Greska: neispravan unos.
```

Primer 5

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n brojeva: 0 1 2 -3 4
 Najvise cifara ima broj 0.
```

Primer 6

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n brojeva: -5 4 -3 2 1
 Najvise cifara ima broj -5.
```

(c) Broj sa najvećom vodećom cifrom. Vodeća cifra je cifra najveće težine u

zapisu broja. Ukoliko ima više takvih, ispisati prvi.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n brojeva: 8 964 -32 511 27
 Broj sa najvecom vodecom cifrom je 964.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 3
 Unesite n brojeva: 0 0 0
 Broj sa najvecom vodecom cifrom je 0.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 3
 Unesite n brojeva: 41 669 -8
 Broj sa najvecom vodecom cifrom je -8.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 0
 Nisu uneti brojevi.
```

Zadatak 1.5.22 Vršena su merenja nadmorskih visina na određenom delu teritorije i naučnike zanima razlika između najveće i najmanje nadmorske visine. Napisati program koji učitava cele brojeve koji predstavljaju nadmorske visine sve do unosa broja nula i ispisuje razliku najveće i najmanje nadmorske visine.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve:
 8 6 5 2 11 7 0
 Razlika: 9
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve:
 8 -1 8 6 0
 Razlika: 9
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve:
 0
 Nisu unete nadmorske
 visine.
```

Primer 4

```
Interakcija sa programom:
 Unesite brojeve:
 -500 0
 Razlika: 0
```

Primer 5

```
Interakcija sa programom:
 Unesite brojeve:
 -500 -300 -5000 0
 Razlika: 4700
```

Zadatak 1.5.23 Napisati program koji učitava ceo broj n ($n > 1$), nenegativan ceo broj d, a zatim i n celih brojeva i izračunava i ispisuje koliko ima parova uzastopnih brojeva među unetim brojevima koji se nalaze na rastojanju d. Rastojanje između brojeva je definisano sa $d(x, y) = |y - x|$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve n i d: 5 2
 Unesite n brojeva: 2 3 5 1 -1
 Broj parova: 2
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve n i d: 5 0
 Unesite n brojeva: 1 1 1 1 1
 Broj parova: 4
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve n i d: 10 5
 Unesite n brojeva:
 -3 6 11 -20 -25 -8 42 37 1 6
 Broj parova: 4
```

Primer 4

```
Interakcija sa programom:
 Unesite brojeve n i d: 1 3
 Greska: neispravan unos.
```

Zadatak 1.5.24 Napisati program koji uneti pozitivan ceo broj transformiše tako što svaku parnu cifru u zapisu broja uveća za jedan. Ispisati dobijeni broj. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 2417
 Rezultat: 3517
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 138
 Rezultat: 139
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 59
 Rezultat: 59
```

Zadatak 1.5.25 Napisati program koji učitava jedan ceo broj i zatim formira i ispisuje broj koji se dobija izbacivanjem svake druge cifre polaznog broja, idući sa desna na levo.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 21854
 Rezultat: 284
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: -18
 Rezultat: -8
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 1
 Rezultat: 1
```

*** Zadatak 1.5.26** Napisati program koji na osnovu unetog pozitivnog celog broja formira i ispisuje broj koji se dobija izbacivanjem cifara koje su u polaznom broju jednake zbiru svojih suseda. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 28631
 Rezultat: 2631
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 440
 Rezultat: 40
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -5
 Greska: neispravan unos.
```

*** Zadatak 1.5.27** Broj je *palindrom* ukoliko se isto čita i sa leve i sa desne strane. Napisati program koji učitava pozitivan ceo broj i proverava da li je učitani broj palindrom. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 25452
 Broj je palindrom.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 895
 Broj nije palindrom.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 5
 Broj je palindrom.
```

Zadatak 1.5.28 Fibonačijev niz počinje članovima 0 i 1, a svaki naredni član se dobija kao zbir prethodna dva. Napisati program koji učitava nenegativan ceo broj n i određuje i ispisuje n-ti član Fibonačijevog niza. Niz se indeksira počevši od nule. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 10
 F[10] = 55
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: -100
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 40
 F[40] = 102334155
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 20
 F[20] = 6765
```

Zadatak 1.5.29 Niz prirodnih brojeva formira se prema sledećem pravilu:

$$a_{n+1} = \begin{cases} \frac{a_n}{2} & \text{ako je } a_n \text{ parno} \\ \frac{3 \cdot a_n + 1}{2} & \text{ako je } a_n \text{ neparno} \end{cases}$$

Napisati program koji za uneti početni član niza a_0 (pozitivan ceo broj) štampa niz brojeva sve do onog člana niza koji je jednak 1. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite prvi clan: 56
 Clanovi niza:
 56 28 14 7 11 17 26 13
 20 10 5 8 4 2 1
```

Primer 2

```
Interakcija sa programom:
 Unesite prvi clan: -48
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite prvi clan: 67
 Clanovi niza:
 67 101 152 76 38 19 29
 44 22 11 17 26 13 20 10
 5 8 4 2 1
```

*** Zadatak 1.5.30** Papir A_0 ima površinu $1m^2$ i odnos stranica $1 : \sqrt{2}$. Papir A_1 dobija se podelom papira A_0 po dužoj ivici. Papir A_2 dobija se podelom A_1 papira po dužoj ivici itd. Napisati program koji za uneti nenegativan broj k ispisuje dimenzije papira A_k u milimetrima. Rezultat ispisati kao celobrojne vrednosti. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite format papira: 4
 Dimenzije papira: 210 297
```

Primer 2

```
Interakcija sa programom:
 Unesite format papira: 0
 Dimenzije papira: 840 1189
```

Primer 3

```
Interakcija sa programom:
 Unesite format papira: -7
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite format papira: 9
 Dimenzije papira: 37 52
```

Zadatak 1.5.31 Napisati program koji učitava karaktere dok se ne unese karakter tačka, i ako je karakter malo slovo ispisuje odgovarajuće veliko, ako je karakter veliko slovo ispisuje odgovarajuće malo, a u suprotnom ispisuje isti karakter kao i uneti.

Primer 1

```
Interakcija sa programom:
 Danas je Veoma Lep DAN.
 dANAS JE vEOMA lEP dan
```

Primer 2

```
Interakcija sa programom:
 PROGRAMIRANJE 1 je zanimljivo!.
 programiranje 1 JE ZANIMLJIVO!
```

Zadatak 1.5.32 Napisati program koji učitava karaktere sve do kraja ulaza, a potom ispisuje broj velikih slova, broj malih slova, broj cifara, broj belina i zbir unetih cifara.

Primer 1

```
Interakcija sa programom:
 Unesite tekst:
 Beograd - Nis 230km
 Uzice - Cacak 56.3km
 Subotica - Ruma 139km
 Velika slova: 6
 Mala slova: 32
 Cifre: 9
 Beline: 12
 Suma cifara: 32
```

Primer 2

```
Interakcija sa programom:
 Unesite tekst:
 Isli smo u Afriku da sadimo papriku.
 Velika slova: 2
 Mala slova: 27
 Cifre: 0
 Beline: 7
 Suma cifara: 0
```

Zadatak 1.5.33 Program učitava pozitivan ceo broj n, a potom i n karaktera. Za svaki od samoglasnika ispisati koliko puta se pojavio među unetim karakterima. Ne praviti razliku između malih i velikih slova. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite n karaktera:
 uAbao
 Samoglasnik a: 2
 Samoglasnik e: 0
 Samoglasnik i: 0
 Samoglasnik o: 1
 Samoglasnik u: 1
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 7
 Unesite n karaktera:
 jk+EEae
 Samoglasnik a: 1
 Samoglasnik e: 3
 Samoglasnik i: 0
 Samoglasnik o: 0
 Samoglasnik u: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -7
 Greska: neispravan unos.
```

Zadatak 1.5.34 Program učitava pozitivan ceo broj n, a zatim i n karaktera. Napisati program koji proverava da li se od unetih karaktera može napisati reč *Zima*. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 4
 Unestite 1. karakter:  +
 Unestite 2. karakter:  o
 Unestite 3. karakter:  Z
 Unestite 4. karakter:  j
 Ne moze se napisati rec Zima.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 10
 Unestite 1. karakter:  i
 Unestite 2. karakter:  9
 Unestite 3. karakter:  0
 Unestite 4. karakter:  p
 Unestite 5. karakter:  a
 Unestite 6. karakter:  Z
 Unestite 7. karakter:  o
 Unestite 8. karakter:  m
 Unestite 9. karakter:  M
 Unestite 10. karakter:  -
 Moze se napisati rec Zima.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Zadatak 1.5.35 Napisati program koji učitava pozitivan ceo broj n i ispisuje vrednost sume kubova brojeva od 1 do n, odnosno $s = 1 + 2^3 + 3^3 + \ldots + n^3$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 14
 Suma kubova: 11025
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 25
 Suma kubova: 105625
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Zadatak 1.5.36 Napisati program koji učitava pozitivan ceo broj n i ispisuje sumu kubova, $s = 1+2^3+3^3+\ldots+k^3$, za svaku vrednost $k = 1, \ldots, n$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 [k=1] Suma kubova: 1
 [k=2] Suma kubova: 9
 [k=3] Suma kubova: 36
 [k=4] Suma kubova: 100
 [k=5] Suma kubova: 225
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 8
 [k=1] Suma kubova: 1
 [k=2] Suma kubova: 9
 [k=3] Suma kubova: 36
 [k=4] Suma kubova: 100
 [k=5] Suma kubova: 225
 [k=6] Suma kubova: 441
 [k=7] Suma kubova: 784
 [k=8] Suma kubova: 1296
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -5
 Greska: neispravan unos.
```

Zadatak 1.5.37 Napisati program koji učitava realan broj x i pozitivan ceo broj n i izračunava i ispisuje sumu $S = x + 2 \cdot x^2 + 3 \cdot x^3 + \ldots + n \cdot x^n$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 2 3
 S = 34.000000
```

Primer 2

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 1.5 5
 S = 74.343750
```

Primer 3

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 5.5 0
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite redom brojeve x i n: -0.5 -5
 Greska: neispravan unos.
```

Zadatak 1.5.38 Napisati program koji učitava realan broj x i pozitivan ceo broj n i izračunava i ispisuje sumu $S = 1 + \frac{1}{x} + \frac{1}{x^2} + \ldots \frac{1}{x^n}$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 2 4
 S = 1.937500
```

Primer 2

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 1.8 6
 S = 2.213249
```

Primer 3

```
Interakcija sa programom:
 Unesite redom brojeve x i n: 5.5 0
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite redom brojeve x i n: -0.5 -5
 Greska: neispravan unos.
```

*** Zadatak 1.5.39** Napisati program koji učitava realne brojeve x i *eps* i sa tačnošću *eps* izračunava i ispisuje sumu $S = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$. Za sumu S se kaže da je izračunata sa tačnošću *eps* ako je apsolutna vrednost poslednjeg člana

sume manja od *eps*. UPUTSTVO: *Prilikom računanja sume koristiti prethodni izračunati član sume u računanju sledećeg člana sume. Naime, ako je izračunat član sume* $\frac{x^n}{n!}$ *na osnovu njega se lako može dobiti član* $\frac{x^{n+1}}{(n+1)!}$. *Nikako ne računati stepen i faktorijel odvojeno zbog neefikasnosti takvog rešenja i zbog mogućnosti prekoračenja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite x: 2
 Unesite tacnost eps: 0.001
 S = 7.388713
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite x: 3
 Unesite tacnost eps: 0.01
 S = 20.079666
```

*** Zadatak 1.5.40** Napisati program koji učitava realne brojeve x i *eps* i sa zadatom tačnošću *eps* izračunava i ispisuje sumu $S = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} \ldots$. NAPOMENA: *Voditi računa o efikasnosti rešenja i o mogućnosti prekoračenja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite x: 3
 Unesite tacnost eps: 0.000001
 S = 0.049787
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite x: 3.14
 Unesite tacnost eps: 0.01
 S = 0.049072
```

Zadatak 1.5.41 Napisati program koji učitava realan broj x i pozitivan ceo broj n i izračunava proizvod $P = (1 + \cos(x)) \cdot (1 + \cos(x^2)) \cdot \ldots \cdot (1 + \cos(x^n))$. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Voditi računa o efikasnosti rešenja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite redom brojeve x i n: 3.4 5
 P = 0.026817
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite redom brojeve x i n: 12 8
 P = 2.640565
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite redom brojeve x i n: 12 0
 Greska: neispravan unos.
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite redom brojeve x i n: 12 -6
 Greska: neispravan unos.
```

*** Zadatak 1.5.42** Napisati program koji učitava pozitivan ceo broj n i ispisuje vrednost razlomka

$$\cfrac{1}{1+\cfrac{1}{2+\cfrac{1}{3+\cfrac{1}{4+\cfrac{1}{\ldots+\cfrac{1}{(n-1)+\frac{1}{n}}}}}}}.$$

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 4
 R = 0.697674
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 20
 R = 0.697775
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

*** Zadatak 1.5.43** Napisati program koji učitava realan broj x i pozitivan ceo broj n i računa sumu

$$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!}.$$

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. Napomena: *Voditi računa o efikasnosti rešenja i o mogućnosti prekoračenja.*

Primer 1

```
Interakcija sa programom:
 Unesite x i n: 5.6 8
 S = 0.779792
```

Primer 2

```
Interakcija sa programom:
 Unesite x i n: 14.32 11
 S = -6714.066406
```

Primer 3

```
Interakcija sa programom:
 Unesite x i n: 2 -6
 Greska: neispravan unos.
```

*** Zadatak 1.5.44** Napisati program koji učitava pozitivan ceo broj n i koji računa proizvod

$$P = (1 + \frac{1}{2!})(1 + \frac{1}{3!}) \ldots (1 + \frac{1}{n!}).$$

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. Napomena: *Voditi računa o efikasnosti rešenja i o mogućnosti prekoračenja.*

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 P = 1.838108
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 7
 P = 1.841026
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 10
 P = 1.841077
```

*** Zadatak 1.5.45** Napisati program koji učitava neparan ceo broj n ($n \geq 5$) i izračunava i ispisuje sumu

$$S = 1 \cdot 3 \cdot 5 - 1 \cdot 3 \cdot 5 \cdot 7 + 1 \cdot 3 \cdot 5 \cdot 7 \cdot 9 - 1 \cdot 3 \cdot 5 \cdot 7 \cdot 9 \cdot 11 + \ldots (-1)^{\frac{n-1}{2}} \cdot 1 \cdot 3 \cdot \ldots \cdot n.$$

U slučaju greške pri unosu podataka ispisati odgovarajuću poruku. NAPOMENA: *Voditi računa o efikasnosti rešenja i o mogućnosti prekoračenja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 9
 S = 855
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 11
 S = -9540
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 20
 Greska: neispravan unos.
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: -3
 Greska: neispravan unos.
```

Zadatak 1.5.46 Napisati program koji učitava realne brojeve x i a i pozitivan ceo broj n i zatim izračunava i ispisuje vrednost izraza

$$((\ldots \underbrace{(((x+a)^2+a)^2+a)^2+\ldots a)^2}_{n}.$$

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve x i a: 3.2 0.2
 Unesite broj n: 5
 Izraz = 135380494030332048.000000
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve x i a: 2 1
 Unesite broj n: 3
 Izraz = 10201.000000
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve x i a: 2.6 0.3
 Unesite broj n: 3
 Izraz = 5800.970129
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve x i a: 5.4 7
 Unesite broj n: -2
 Greska: neispravan unos.
```

Zadatak 1.5.47 Napisati program koji za unetu pozitivnu celobrojnu vrednost n ispisuje odgovarajuće brojeve. NAPOMENA: *Pretpostaviti da je unos ispravan.*

(a) Ispisati tablicu množenja.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 1
 1
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 2
 1  2
 2  4
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 4
 1  2  3  4
 2  4  6  8
 3  6  9  12
 4  8  12 16
```

(b) Ispisati sve brojeve od 1 do n^2 (po n brojeva u jednoj vrsti).

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 1  2  3
 4  5  6
 7  8  9
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
 1   2   3   4
 5   6   7   8
 9   10  11  12
 13  14  15  16
```

(c) Ispisati tablicu brojeva tako da su u prvoj vrsti svi brojevi od 1 do n, a svaka naredna vrsta dobija se rotiranjem prethodne vrste za jedno mesto u levo.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 1 2 3
 2 3 1
 3 1 2
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
 1 2 3 4
 2 3 4 1
 3 4 1 2
 4 1 2 3
```

(d) Ispisati pravougli „trougao" sačinjen od „koordinata" svojih tačaka. „Koordinata" tačke je oblika (i, j) pri čemu $i,\ j = 0, \ldots, n$. Prav ugao se nalazi u gornjem levom uglu slike i njegova koordinata je $(0, 0)$. Koordinata i se uvećava po vrsti, a koordinata j po koloni, pa je zato koordinata tačke koja je ispod tačke $(0, 0)$ jednaka $(1, 0)$, a koordinata tačke koja je desno od tačke $(0, 0)$ jednaka $(0, 1)$.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 1
 (0,0)
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 2
 (0,0) (0,1)
 (1,0)
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 4
 (0,0) (0,1) (0,2) (0,3)
 (1,0) (1,1) (1,2)
 (2,0) (2,1)
 (3,0)
```

Zadatak 1.5.48 Napisati program koji za uneti pozitivan ceo broj n zvezdicama iscrtava odgovarajuću sliku. NAPOMENA: *Pretpostaviti da je unos ispravan.*

(a) Slika predstavlja kvadrat stranice n sastavljen od zvezdica.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
 ***
 ***
 ***
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 ****
 ****
 ****
 ****
```

(b) Slika predstavlja rub kvadrata dimenzije n.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 5
 *****
 *   *
 *   *
 *   *
 *****
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 2
 **
 **
```

(c) Slika predstavlja rub kvadrata dimenzije n koji i na glavnoj dijagonali ima zvezdice.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 5
 *****
 **  *
 * * *
 *  **
 *****
```

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 ****
 ** *
 * **
 ****
```

*** Zadatak 1.5.49** Napisati program koji za uneti pozitivan ceo broj n zvezdicama iscrtava slovo X dimenzije n. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 5
 *   *
  * *
   *
  * *
 *   *
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
 * *
  *
 * *
```

*** Zadatak 1.5.50** Napisati program koji za uneti neparan pozitivan broj n korišćenjem znaka + iscrtava veliko plus dimenzije n. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 5
    +
    +
 +++++
    +
    +
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
  +
 +++
  +
```

Zadatak 1.5.51 Napisati program koji učitava pozitivan ceo broj n, a potom iscrtava odgovarajuću sliku. NAPOMENA: *Pretpostaviti da je unos ispravan.*

(a) Slika predstavlja pravougli trougao sastavljen od zvezdica. Kateta trougla je dužine n, a prav ugao se nalazi u gornjem levom uglu slike.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
 ***
 **
 *
```

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 ****
 ***
 **
 *
```

(b) Slika predstavlja pravougli trougao sastavljen od zvezdica. Kateta trougla je dužine n, a prav ugao se nalazi u donjem levom uglu slike.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
 *
 **
 ***
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 *
 **
 ***
 ****
```

(c) Slika predstavlja pravougli trougao sastavljen od zvezdica. Kateta trougla je dužine n, a prav ugao se nalazi u gornjem desnom uglu slike.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
 ***
  **
   *
```

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 ****
  ***
   **
    *
```

(d) Slika predstavlja pravougli trougao sastavljen od zvezdica. Kateta trougla je dužine n, a prav ugao se nalazi u donjem desnom uglu slike.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
   *
  **
 ***
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
    *
   **
  ***
 ****
```

(e) Slika predstavlja trougao sastavljen od zvezdica. Trougao se dobija spajanjem dva pravougla trougla kateta dužine n, pri čemu je prav ugao prvog trougla u njegovom donjem levom uglu, dok je prav ugao drugog trougla u njegovom gornjem levom uglu, a spajanje se vrši po horiznotalnoj kateti.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 *
 **
 ***
 **
 *
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
 *
 **
 ***
 ****
 ***
 **
 *
```

(f) Slika predstavlja rub jednakokrakog pravouglog trougla čije su katete dužine n. Program učitava karakter c i taj karakter koristi za iscrtavanje ruba trougla.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 4
 Unesite karakter c: *
 *
 **
 * *
 ****
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 5
 Unesite karakter c: +
 +
 ++
 + +
 +  +
 +++++
```

Zadatak 1.5.52 Napisati program koji učitava pozitivan ceo broj n, a potom iscrtava odgovarajuću sliku. Napomena: *Pretpostaviti da je unos ispravan.*

(a) Slika predstavlja jednakostranični trougao stranice n koji je sastavljen od zvezdica.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
   *
  ***
 *****
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
    *
   ***
  *****
 *******
```

(b) Slika predstavlja jednakostranični trougao stranice n koji je sastavljen od zvezdica pri čemu je vrh trougla na dnu slike.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 *****
  ***
   *
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 4
 *******
  *****
   ***
    *
```

(c) Slika predstavlja trougao koji se dobija spajanjem dva jednakostranična trougla stranice n koji su sastavljeni od zvezdica.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
   *
  ***
 *****
  ***
   *
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 5
     *
    ***
   *****
  *******
 *********
  *******
   *****
    ***
     *
```

(d) Slika predstavlja rub jednakostraničnog trougla čija stranica je dužine n.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
   *
  * *
 * * *
```

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
     *
    * *
   *   *
  *     *
 * * * * *
```

(e) Slika se dobija spajanjem dva jednakostranična trougla čija stranica je dužine n. Iscrtavati samo rub trouglova.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
   *
  * *
 * * *
  * *
   *
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 5
     *
    * *
   *   *
  *     *
 * * * * *
  *     *
   *   *
    * *
     *
```

*** Zadatak 1.5.53** Napisati program koji za uneti pozitivan ceo broj n iscrtava strelice dimenzije n. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 *
  *
 ***
  *
 *
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 5
 *
  *
   *
    *
 *****
    *
   *
  *
 *
```

*** Zadatak 1.5.54** Napisati program koji učitava pozitivan ceo broj n i iscrtava sliku koja se dobija na sledeći način: u prvom redu je jedna zvezdica, u drugom redu su dve zvezdice razdvojene razmakom, treći red je sastavljen od zvezdica i iste je dužine kao i drugi red, četvrti red se sastoji od tri zvezdice razdvojene razmakom, a peti red je sastavljen od zvezdica i iste je dužine kao i četvrti red itd. Ukupna visina slike je n. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 7
 *
 * *
 ***
 * * *
 *****
 * * * *
 *******
```

*** Zadatak 1.5.55** Napisati program koji učitava pozitivne cele brojeve m i n i iscrtava jedan do drugog n kvadrata čija je svaka stranica sastavljena od m zvezdica razdvojenih prazninom. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve n i m: 5 3
 * * * * * * * * * * * * * * *
 *       *       *       *
 *       *       *       *
 *       *       *       *
 * * * * * * * * * * * * * * *
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite brojeve n i m: 4 4
 * * * * * * * * * * * * * * * *
 *     *     *     *     *
 *     *     *     *     *
 * * * * * * * * * * * * * * * *
```

*** Zadatak 1.5.56** Napisati program koji učitava pozitivan ceo broj n i iscrtava romb sastavljen od minusa u pravougaoniku sastavljenom od zvezdica. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 6
 ************
 *****--*****
 ****----****
 ***------***
 **--------**
 *----------*
 **--------**
 ***------***
 ****----****
 *****--*****
 ************
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 2
 ****
 *--*
 ****
```

Zadatak 1.5.57 Napisati program koji učitava ceo broj n ($n \geq 2$) i koji iscrtava sliku kuće sa krovom: kuća je kvadrat stranice n, a krov jednakostranični trougao stranice n. Pretpostaviti da je unos korektan.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
    *
   * *
  *   *
 * * * *
 *     *
 *     *
 * * * *
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 3
   *
  * *
 * * *
 *   *
 * * *
```

*** Zadatak 1.5.58** Napisati program koji učitava pozitivan ceo broj n i ispisuje brojeve od 1 do n, zatim od 2 do $n-1$, 3 do $n-2$, itd. Ispis se završava kada nije moguće ispisati ni jedan broj. NAPOMENA: *Pretpostaviti da je unos*

ispravan.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 1 2 3 4 5 2 3 4 3
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 6
 1 2 3 4 5 6 2 3 4 5 3 4
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 7
 1 2 3 4 5 6 7 2 3 4 5 6 3 4 5 4
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 3
 1 2 3 2
```

*** Zadatak 1.5.59** Napisati program koji učitava pozitivan ceo broj n i ispisuje izabrane brojeve u n redova. U prvom redu, program ispisuje sve brojeve iz intervala $[1, n]$. U drugom redu, program ispisuje svaki drugi broj iz ovog intervala. U trećem redu, program ispisuje svaki treći broj iz ovog intervala, i tako redom. Na kraju, u n-tom redu, program ispisuje samo broj 1. Napomena: *Pretpostaviti da je unos ispravan.*

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 3
 1 2 3
 1 3
 1
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 7
 1 2 3 4 5 6 7
 1 3 5 7
 1 4 7
 1 5
 1 6
 1 7
 1
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 1
 1
```

1.6 Rešenja

Rešenje 1.5.1

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebne promenljive. */
  int i;

  /* Promenljiva i kontrolise koliko puta ce se petlja izvrsiti i
     naziva se brojac petlje. Njena pocetna vrednost se postavlja
     na 0 jer se u pocetku petlja nije ni jednom izvrsila. */
  i = 0;

  /* Petlja ce se izvrsiti za i=0,1,2,3,4. Kada i dostigne vrednost
     5 uslov i < 5 nece biti ispunjen i prelazi se na prvu sledecu
     naredbu nakon tela petlje. */
  while (i < 5) {
    /* Ispis poruke. */
    printf("Mi volimo da programiramo.\n");

    /* Uvecavanje brojaca za 1. */
    i++;
  }

  return 0;
}
```

Rešenje 1.5.2

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int i, n;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: pogresan unos broja n.\n");
    return 1;
  }

  /* Inicijalizacija brojaca. */
  i = 0;

```

```
  /* Trazena poruka se ispisuje n puta. */
  while (i < n) {
    printf("Mi volimo da programiramo.\n");
    i++;
  }

  return 0;
}
```

Rešenje 1.5.3

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int i, n;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n < 0) {
    printf("Greska: pogresan unos broja n.\n");
    return 1;
  }

  /* Inicijalizacija brojaca. */
  i = 0;

  /* Posto je potrebno ispisati sve brojeve [0,n], telo petlje se
     izvrsava za svako i <= n. */
  while (i <= n) {
    /* Ispis trenutne vrednosti brojaca. */
    printf("%d\n", i);

    /* Prelazak na sledeci broj.  */
    i++;
  }

  return 0;
}
```

Rešenje 1.5.4

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, m, i;
```

```
7   /* Ucitavanje vrednosti granica intervala. */
    printf("Unesite granice intervala: ");
9   scanf("%d%d", &n, &m);

11  /* Provera ispravnosti ulaznih podataka. */
    if (m < n) {
13    printf("Greska: pogresan unos granica.\n");
      return 1;
15  }

17  /* a) I nacin: Koriscenjem while petlje. */
    /* Inicijalizacija brojaca na levu granicu intervala. */
19  i = n;

21  /* Ispis svih vrednosti brojaca izmedju leve i desne granice
       intervala, ukljucujuci i same granice. */
23  while (i <= m) {
      printf("%d ", i);
25    i++;
    }
27
    /* b) II nacin: Koriscenjem for petlje.
29
       Naredba i=n se izvrsava jednom, pre prve iteracije. Uslov
31     petlje i<=m se proverava pre svake iteracije. Naredba i++ se
       izvrsava nakon svake iteracije.
33
       for (i = n; i <= m; i++){
35       printf("%d ", i);
       } */
37
    /* c) III nacin: Koriscenjem do while petlje.
39
       Uslov petlje se proverava na kraju svake iteracije. Zbog toga
41     se do while petlja izvrsava bar jednom, cak i u slucaju da
       uslov petlje nikada nije ispunjen. U ovom slucaju je to
43     ispravno jer je poznato da ce interval imati bar jedan
       element. U opstem slucaju to ne mora da vazi.
45
       i = n;
47     do {
         printf("%d ", i);
49       i++;
       } while (i <= m); */
51
    printf("\n");
53
    return 0;
55  }
```

Rešenje 1.5.5

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, tekuca_vrednost;

  /* Za cuvanje vrednosti faktorijela se koristi tip unsigned long
     jer izracunata vrednost moze da bude jako veliki broj. */
  unsigned long faktorijel;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n < 0) {
    printf("Greska: neispravan unos..\n");
    return 1;
  }

  if (n >= 22) {
    printf("Pri racunanju %d! ce doci do prekoracenja.\n", n);
    return 1;
  }

  /* Tekuca vrednost uzima vrednosti n, n-1, n-2, ..., 2. Na
     pocetku se inicijalizuje na n, a zatim se u svakoj iteraciji
     umanjuje za 1. */
  tekuca_vrednost = n;

  /* Inicijalizacija vrednosti faktorijela. */
  faktorijel = 1;

  /* Racunanje vrednosti faktorijela mnozenjem trenutnog rezultata
     promenljivom cija vrednost krece od n, a zatim se u svakoj
     iteraciji umanjuje za 1. */
  while (tekuca_vrednost > 1) {
    faktorijel = faktorijel * tekuca_vrednost;
    tekuca_vrednost--;
  }

  /* Ispis rezultata. */
  printf("%d! = %lu\n", n, faktorijel);

  return 0;
}
```

Rešenje 1.5.6

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  float x, rezultat;

  /* Ucitavanje vrednosti brojeva x i n. */
  printf("Unesite redom brojeve x i n: ");
  scanf("%f %d", &x, &n);

  /* Provera ispravnosti ulaza. */
  if (n < 0) {
    printf("Greska: neispravan unos broja n.\n");
    return 1;
  }

  /* Inicijalizacija rezultata. */
  rezultat = 1;

  /* Vrednost n-tog stepena broja x se dobija tako sto se tekuca
     vrednost rezultata n puta pomnozi brojem x.
     (rezultat = x * x * ... * x) = x^n */
  for (i = 0; i < n; i++)
    rezultat = rezultat * x;

  /* Ispis rezultata. */
  printf("Rezultat: %.5f\n", rezultat);

  return 0;
}
```

Rešenje 1.5.7

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i, znak;
  float x, rezultat;

  /* Ucitavanje vrednosti brojeva x i n. */
  printf("Unesite redom brojeve x i n: ");
  scanf("%f %d", &x, &n);

  /* Pamti se znak stepena i izracunava se apsolutna vrednost stepena
    . */
  znak = 1;
```

```
15  if (n < 0) {
      znak = -1;
17    n = abs(n);
    }
19
    /* Inicijalizacija rezultata. */
21  rezultat = 1;

23  /* Racunanje vrednosti x^n. */
    for (i = 0; i < n; i++)
25    rezultat = rezultat * x;

27  /* Ako je stepen bio negativan, rezultat je 1/x^n. */
    if (znak == -1)
29    printf("Rezultat: %.3f\n", 1 / rezultat);
    else
31    printf("Rezultat: %.3f\n", rezultat);

33  return 0;
}
```

Rešenje 1.5.8

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracija potrebnih promenljivih. */
5    int n, i;

7    /* Ucitavanje vrednosti broja n. */
     printf("Unesite broj n: ");
9    scanf("%d", &n);

11   /* Provera ispravnosti ulaza. */
     if (n <= 0) {
13     printf("Greska: neispravan unos.\n");
       return 1;
15   }

17   printf("Pravi delioci:\n");
     /* I nacin: Za svaki broj iz intervala [2, n-1] se proverava da
19      li deli broj n (tj. da li je ostatak pri deljenju sa n jednak
        nuli). Ako je uslov ispunjen, taj broj se ispisuje.
21      for (i = 2; i < n; i++)
          if (n % i == 0)
23          printf("%d ", i);

25      printf("\n"); */

27   /* II nacin (brzi): Provera se ne vrsi za sve brojeve iz
        intervala [2, n-1], vec za brojeve iz intervala [2, sqrt(n)],
```

```
29      tj. za sve brojeve k za koje vazi da je k*k <= n. */
    for (i = 2; i * i <= n; i++) {
31    /* Ako i deli n, treba razlikovati dva slucaja. */
      if (n % i == 0) {
33      if (i == n / i) {
          /* I slucaj: kada je i koren broja, ispisuje se samo broj i,
35           npr. za n = 16, i = 4, ispisuje se samo 4. */
          printf("%d ", i);
37      } else {
          /* II slucaj: kada i nije koren broja, ispisuje se i broj
39           i i broj n/i, npr. za n = 16, i = 2 ispisuju se i 2 i 8.
             */
41        printf("%d %d ", i, n / i);
        }
43    }
    }
45  printf("\n");

47  return 0;
}
```

Rešenje 1.5.9

```
1 #include <stdio.h>

3 int main() {
    /* Deklaracija potrebne promenljive. */
5   int n;

7   /* Ucitavanje vrednosti broja n. */
    printf("Unesite broj: ");
9   scanf("%d", &n);

11  /* Slucaj kada broj n ima vrednost nula se posebno obradjuje.
       U suprotnom bi se petlja u nastavku beskonacno izvrsavala. */
13  if (n == 0) {
      printf("0\n");
15    return 0;
    }
17
    /* Uklanjanje poslednje cifre broja se vrsi deljenjem broja sa 10.
19     Ovaj postupak se ponavlja sve dok je poslednja cifra nula. */
    while (n % 10 == 0)
21    n = n / 10;

23  /* Ispis rezultata. */
    printf("Rezultat: %d\n", n);
25
    return 0;
27 }
```

Rešenje 1.5.10

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracija potrebne promenljive. */
  int x;

  /* Ucitavanje vrednosti broja x. */
  printf("Unesite ceo broj:");
  scanf("%d", &x);

  /* Izracunava se apsolutna vrednost broja da bi izdvojene cifre
    bile
     pozitivni brojevi. Na primer, 123%10 je 3, a -123%10 je -3. */
  x = abs(x);

  /* Slucaj kada je uneti broj 0 se posebno obradjuje. */
  if (x == 0) {
    printf("0\n");
    return 0;
  }

  /* U petlji se obradjuje cifra po cifra broja, dok god ima
     neobradjenih cifara u broju. */
  printf("Rezultat: ");
  while (x != 0) {
    /* Ispis poslednje cifre broja x. */
    printf("%d ", x % 10);

    /* Uklanjanje poslednje cifre broja x. */
    x /= 10;
  }
  printf("\n");

  return 0;
}
```

Rešenje 1.5.11

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, suma, n_kopija;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj: ");
  scanf("%d", &n);
```

```
  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Pravljenje kopije originalnog broja, da bi originalna vrednost n
     ostala nepromenjena. */
  n_kopija = n;

  /* Inicijalizacija sume cifara. */
  suma = 0;

  /* Racunanje sume cifara. */
  while (n_kopija != 0) {
    /* Dodavanje poslednje cifre na sumu. */
    suma += n_kopija % 10;
    /* Uklanjanje poslednje cifre. */
    n_kopija /= 10;
  }

  /* Ispis rezultata. */
  if (n % suma == 0)
    printf("Broj %d je deljiv sa %d.\n", n, suma);
  else
    printf("Broj %d nije deljiv sa %d.\n", n, suma);

  return 0;
}
```

Rešenje 1.5.12

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int t, x, i;
  int ukupan_prihod, ukupan_rashod, ukupan_rashod_abs;

  /* Ucitavanje vrednosti broja t. */
  printf("Unesite broj t:");
  scanf("%d", &t);

  /* Provera ispravnosti ulaza. */
  if (t < 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  } else if (t == 0) {
    printf("Nema evidentiranih transakcija.");
    return 0;
```

```
  }

  /* Inicijalizacija suma. */
  ukupan_prihod = 0;
  ukupan_rashod = 0;

  /* Ucitavanje transakcija i racunanje suma. */
  printf("Unesite transakcije: ");
  i = 0;
  while (i < t) {
    /* Ucitavanje jedne transakcije. */
    scanf("%d", &x);

    /* Dodavanje ucitane vrednosti na odgovarajucu sumu. */
    if (x < 0)
      ukupan_rashod += x;
    else
      ukupan_prihod += x;

    /* Uvecavanje brojaca. */
    i++;
  }

  /* Ispis rezultata. */
  printf("Prihod: %d\n", ukupan_prihod);
  printf("Rashod: %d\n", ukupan_rashod);

  ukupan_rashod_abs = abs(ukupan_rashod);
  if (ukupan_prihod >= ukupan_rashod_abs)
    printf("Zarada: %d\n", ukupan_prihod - ukupan_rashod_abs);
  else
    printf("Gubitak: %d\n", ukupan_rashod_abs - ukupan_prihod);

  return 0;
}
```

Rešenje 1.5.13

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, x, i;
  int zbir = 0;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
```

```
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ucitavanje brojeva i racunanje trazenog zbira. */
  printf("Unesite n brojeva: ");
  i = 0;
  while (i < n) {
    /* Ucitavanje jednog broja. */
    scanf("%d", &x);

    /* Ako je ucitani broj negativan i neparan, dodaje se na
       zbir. */
    if (x < 0 && x % 2 != 0)
      zbir = zbir + x;

    /* Uvecavanje brojaca. */
    i++;
  }

  /* Ispis rezultata. */
  printf("Zbir neparnih i negativnih: %d\n", zbir);

  return 0;
}
```

Rešenje 1.5.14

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int x, proizvod;

  /* Indikator koji oznacava da li je korisnik uneo bar jedan
     broj. */
  int unet_bar_jedan = 0;

  /* Indikator koji oznacava da li je korisnik uneo bar jedan
     pozitivan broj. */
  int unet_pozitivan = 0;

  /* Inicijalizacija proizvoda. */
  proizvod = 1;

  printf("Unesite brojeve:");
  /* Petlja ciji je uslov uvek ispunjen. */
  while (1) {
    /* Ucitavanje jednog broja. */
    scanf("%d", &x);
```

```
24      /* Ako je uneta nula, petlja se prekida naredbom break. */
        if (x == 0)
26        break;

28      /* Ako petlja nije prekinuta, znaci da je unet bar jedan broj.
           Iz tog razloga se vrednost indikatora za unete brojeve
30         postavlja na 1. */
        unet_bar_jedan = 1;
32
        /* Provera da li je broj x pozitivan. */
34      if (x > 0) {
          /* Ako jeste, znaci da je unet bar jedan pozitivan broj i iz
36           tog razloga se vrednost odgovarajuceg indikatora postavlja
             na 1. */
38        unet_pozitivan = 1;

40        /* Azuriranje vrednosti proizvoda pozitivnih brojeva. */
          proizvod = proizvod * x;
42      }
    }
44
    /* Ispis rezultata. */
46  if (unet_bar_jedan == 0)
      printf("Nije unet nijedan broj.\n");
48  else if (unet_pozitivan == 0)
      printf("Medju unetim brojevima nema pozitivnih.\n");
50  else
      printf("Proizvod pozitivnih brojeva je %d.\n", proizvod);
52
    return 0;
54  }
```

Rešenje 1.5.15 Pogledajte zadatak 1.5.10.

Rešenje 1.5.16

```
    #include <stdio.h>
2
    int main() {
4     /* Deklaracija i inicijalizacije potrebnih promenljivih. */
      int x, broj_brojeva = 0, suma = 0;
6
      /* Ucitavanje brojeva sve do unosa broja 0. */
8     printf("Unesite brojeve: ");
      while (1) {
10      scanf("%d", &x);

12      if (x == 0)
          break;
14
        /* Dodavanje ucitanog broja na sumu. */
```

```
16      suma += x;

18      /* Uvecavanje broja ucitanih brojeva. */
        broj_brojeva++;
20    }

22    /* Ispis rezultata. Napomena: I suma i broj brojeva su celi
         brojevi pa je neophodno bar jednu od te dve vrednosti pretvoriti
24       u realan broj kako deljenje ne bi bilo celobrojno. */
      if (broj_brojeva == 0)
26      printf("Nisu uneti brojevi.\n");
      else
28      printf("Aritmeticka sredina: %.4f\n",
                (double) suma / broj_brojeva);
30
      return 0;
32  }
```

Rešenje 1.5.17

```
    #include <stdio.h>
2
    int main() {
4     /* Deklaracije potrebnih promenljivih. */
      float cena, suma = 0;
6     int broj_artikla = 0;

8     /* Ucitavanje cena sve do unosa broja 0. */
      printf("Unesite cene: ");
10    while (1) {
        scanf("%f", &cena);
12
        if (cena == 0)
14        break;

16      /* Provera ispravnosti ulaza. */
        if (cena < 0) {
18        printf("Greska: neispravan unos cene.\n");
          return 1;
20      }

22      /* Uvecavanje sume za vrednost unete cene. */
        suma += cena;
24
        /* Uvecavanje broja unetih artikala za 1. */
26      broj_artikla++;
      }
28
      /* Ispis rezultata. */
30    if (broj_artikla == 0)
        printf("Nisu unete cene.\n");
```

```
32    else
        printf("Prosecna cena: %.4f\n", suma / broj_artikla);
34
    return 0;
36  }
```

Rešenje 1.5.18

```
1   #include <stdio.h>

3   int main() {
      /* Deklaracije potrebnih promenljivih. */
5     int n, i, broj_promena = 0;
      double prethodni, trenutni;
7
      /* Ucitavanje vrednosti broja n. */
9     printf("Unesite broj n ");
      scanf("%d", &n);
11
      /* Provera ispravnosti ulaza. */
13    if (n <= 0) {
        printf("Greska: neispravan unos.\n");
15      return 1;
      }
17
      printf("Unesite brojeve: ");
19    /* Provera promene znaka se vrsi za svaka dva susedna uneta
         broja. Prvi broj se ucitava pre petlje i smesta se u
21       promenljivu prethodni. Zatim se u petlji ucitava drugi i
         njihov znak se poredi. Postupak se ponavlja za sve parove,
23       tako sto se uvek na kraju petlje poslednji ucitani broj
         postavi da bude prethodni za sledecu iteraciju. */
25    scanf("%lf", &prethodni);

27    /* Kako je vec jedan broj unet, brojac se postavlja na 1, a ne na
         0. */
29    for (i = 1; i < n; i++) {
        /* Ucitavanje broja. */
31      scanf("%lf", &trenutni);

33      /* Provera da li je doslo do promene znaka izmedju prethodnog
           i trenutnog broja. Oni su razlicitog znaka ako vazi:
35         1. da im je proizvod negativan ILI
           2. da im je proizvod nula, a jedan od njih je negativan. */
37      if (prethodni * trenutni < 0)
          broj_promena++;
39      else if (prethodni * trenutni == 0 &&
                 (prethodni < 0 || trenutni < 0))
41        broj_promena++;

43      /* Trenutni broj postaje prethodni za sledecu iteraciju. */
```

```
    prethodni = trenutni;
  }

  /* Ispis rezultata. */
  printf("Broj promena je %d.\n", broj_promena);

  return 0;
}
```

Rešenje 1.5.19

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  float cena, min_cena;

  /* Ucitavanje broja artikala. */
  printf("Unesite broj artikala:");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  printf("Unesite cene artikala:");

  /* Inicijalizacija minimalne cene na vrednost cenu prvog artikla.
     Zbog toga se cena prvog artikla ucitava pre petlje. */
  scanf("%f", &cena);
  if (cena <= 0) {
    printf("Greska: neispravan unos cene.\n");
    return 1;
  }
  min_cena = cena;

  /* Ucitavanje i preostalih n-1 cena i racunanje najmanje. */
  for (i = 1; i < n; i++) {
    scanf("%f", &cena);

    if (cena <= 0) {
      printf("Greska: neispravan unos cene.\n");
      return 1;
    }

    if (cena < min_cena)
      min_cena = cena;
    i++;
```

```
41    }
43    /* Ispis rezultata. */
      printf("Najmanja cena: %f\n", min_cena);
45
      return 0;
47  }
```

Rešenje 1.5.20

```
1   #include <stdio.h>
3   int main() {
      /* Deklaracije potrebnih promenljivih. */
5     float cena, m;
      unsigned int n, i, broj_artikala = 0;
7
      /* Ucitavanje vrednosti budzeta. */
9     printf("Nikolin budzet: ");
      scanf("%f", &m);
11
      /* Ucitavanje broja artikala. */
13    printf("Unesite broj artikala: ");
      scanf("%u", &n);
15
      /* Ucitavanje cena artikala i racunanje rezultata. */
17    printf("Unesite cene artikala: ");
      for (i = 0; i < n; i++) {
19      /* Ucitavanje cene artikla. */
        scanf("%f", &cena);
21
        /* Provera da li Nikola moze da kupi trenutni artikal. */
23      if (cena <= m)
          broj_artikala++;
25    }
27    /* Ispis rezultata. */
      printf("Ukupno artikala: %d\n", broj_artikala);
29
      return 0;
31  }
```

Rešenje 1.5.21

Rešenje (a)

```
1   #include <stdio.h>
    #include <stdlib.h>
3
```

```
int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i, x, rezultat;
  int x_desetica, najveca_desetica;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n < 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ako nema unetih brojeva, ispisuje se odgovarajuca poruka. */
  if (n == 0) {
    printf("Nisu uneti brojevi.\n");
    return 0;
  }

  printf("Unesite brojeve: ");

  /* Prvi broj se ucitava pre petlje zbog ispravne
     inicijalizacije. */
  scanf("%d", &x);
  /* Promenljiva najveca_desetica se postavlja na cifru desetica
     ucitanog broja. Napomena: Pri racunanju se koristi apsolutna
     vrednost broja jer je npr. (-123/10)= -12 i -12 % 10 = -2, a
     cifra desetica treba da bude 2. */
  najveca_desetica = (abs(x) / 10) % 10;
  /* Kako je na kraju potrebno ispisati broj cija je cifra desetica
     najveca, trenutna vrednost rezultata se postavlja na vrednost
     ucitanog broja. */
  rezultat = x;

  /* Ucitavanje preostalih n-1 brojeva i u slucaju nailaska na
     broj cija je cifra desetica veca od trenutno najvece, vrsi se
     azuriranje najvece desetice i rezultata. */
  for (i = 1; i < n; i++) {
    scanf("%d", &x);

    x_desetica = (abs(x) / 10) % 10;

    if (x_desetica > najveca_desetica) {
      najveca_desetica = x_desetica;
      rezultat = x;
    }
  }

  /* II nacin: Inicijalizacija najvece desetice na neku vrednost
     koja je sigurno manja od svih vrednosti koje cifra desetica
```

```
    moze da uzme (dakle, bilo sta sto je manje od 0 jer cifra
57  desetica moze imati vrednosti izmedju 0 i 9). Zatim se u
    petlji izracunava rezultat, analogno prvom nacinu.
59
    najveca_desetica = -1;
61  for(i=0; i<n; i++) {
      scanf("%d", &x);
63    x_desetica = (abs(x) / 10) % 10;
      if (x_desetica > najveca_desetica) {
65      najveca_desetica = x_desetica;
        rezultat = x;
67    }
    } */
69
  /* Ispis rezultata. */
71 printf("Broj sa najvecom cifrom desetica: %d\n", rezultat);

73 return 0;
}
```

Rešenje (b)

```
#include <stdio.h>
2 #include <stdlib.h>

4 int main() {
  /* Deklaracije potrebnih promenljivih. */
6 int n, i;
  int x, x_kopija, broj_cifara;
8 int najveci_broj_cifara, rezultat;

10 /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
12 scanf("%d", &n);

14 /* Provera ispravnosti ulaza. */
  if (n < 0) {
16   printf("Greska: neispravan unos.\n");
     return 1;
18 }

20 /* Ako nema unetih brojeva, ispisuje se odgovarajuca poruka. */
  if (n == 0) {
22   printf("Nisu uneti brojevi.\n");
     return 0;
24 }

26 /* Postavljanje maksimalnog broja cifara na 0. Ovo je ispravno
      jer svaki broj ima vise od 0 cifara. */
28 najveci_broj_cifara = 0;
```

```
  printf("Unesite n brojeva: ");
  for (i = 0; i < n; i++) {
    scanf("%d", &x);

    /* Racunanje broja cifara unetog broja x. */
    x_kopija = abs(x);
    broj_cifara = 0;
    do {
      broj_cifara++;
      x_kopija = x_kopija / 10;
    } while (x_kopija != 0);

    /* Ako je broj cifara unetog broja veci od najveceg broja
       cifara, azuriraju se vrednosti najveceg broja cifara i
       tekuceg rezultata. */
    if (broj_cifara > najveci_broj_cifara) {
      najveci_broj_cifara = broj_cifara;
      rezultat = x;
    }
  }

  /* Ispis rezultata. */
  printf("Najvise cifara ima broj %d.\n", rezultat);

  return 0;
}
```

Rešenje (c)

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  int x, x_kopija, vodeca_cifra;
  int najveca_vodeca_cifra, rezultat;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n < 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ako nema unetih brojeva, ispisuje se odgovarajuca poruka. */
  if (n == 0) {
    printf("Nisu uneti brojevi.\n");
```

```
23      return 0;
    }
25
    /* Inicijalizacija najvece vodece cifre na -1. */
27  najveca_vodeca_cifra = -1;

29  printf("Unesite n brojeva: ");
    for (i = 0; i < n; i++) {
31    scanf("%d", &x);

33    /* Racunanje vodece cifre ucitanog broja x. */
      x_kopija = abs(x);
35    while (x_kopija >= 10) {
        x_kopija = x_kopija / 10;
37    }
      vodeca_cifra = x_kopija;
39
      /* Ako je izdvojena cifra veca od najvece vodece cifre,
41       azuriraju se vrednosti najvece vodece cifre i rezultata. */
      if (vodeca_cifra > najveca_vodeca_cifra) {
43      najveca_vodeca_cifra = vodeca_cifra;
        rezultat = x;
45    }
    }
47
    /* Ispis rezultata. */
49  printf("%d\n", rezultat);

51  return 0;
}
```

Rešenje 1.5.22 Pogledajte zadatak 1.5.19.

Rešenje 1.5.23

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  int main() {
5   /* Deklaracije potrebnih promenljivih. */
    int n, d, i;
7   int x, y, broj_parova = 0;

9   /* Ucitavanje vrednosti n i d. */
    printf("Unesite brojeve n i d: ");
11  scanf("%d %d", &n, &d);

13  /* Provera ispravnosti ulaza. */
    if (n <= 1 || d < 0) {
15    printf("Greska: neispravan unos.\n");
      return 1;
```

```
  }

  printf("Unesite n brojeva: ");

  /* Prvi broj se ucitava pre petlje. */
  scanf("%d", &x);

  for (i = 1; i < n; i++) {
    scanf("%d", &y);

    /* Provera da li su brojevi na rastojanju d. */
    if (abs(y - x) == d)
      broj_parova++;

    /* Broj iz tekuce iteracije se cuva kako bi mogao da se
       upotrebljava u narednoj iteraciji. */
    x = y;
  }

  /* Ispis rezultata. */
  printf("Broj parova: %d\n", broj_parova);

  return 0;
}
```

Rešenje 1.5.24

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, cifra, pozicija;
  unsigned int rezultat;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Inicijalizacija pozicije i rezultata. Pozicija oznacava tezinu
     trenutne cifre i moze imati vrednosti 1, 10, 100, 1000, ... */
  pozicija = 1;
  rezultat = 0;

  /* Izdvajanje cifre po cifre broja sve dok ima neobradjenih
     cifara. */
```

```
  while (n > 0) {
    /* Izdvajanje poslednje cifre iz zapisa. U slucaju da je njena
       vrednost paran broj, izdvojena cifra se uvecava za 1. */
    cifra = n % 10;
    if (cifra % 2 == 0)
      cifra++;

    /* Novi broj se formira tako sto se izdvojena cifra pomnozi
       odgovarajucom tezinom (stepenom) pozicije i doda na tekuci
       rezultat. */
    rezultat += cifra * pozicija;

    /* Uklanjanje poslednje cifre broja. */
    n /= 10;

    /* Mnozenje pozicije sa 10. */
    pozicija *= 10;
  }

  /* Ispis izracunate vrednosti. */
  printf("Rezultat: %d\n", rezultat);

  return 0;
}
```

Rešenje 1.5.25

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int x, pozicija, rezultat, cifra;
  int znak = 1;

  /* Ucitavanje vrednosti polaznog broja. */
  printf("Unesite broj: ");
  scanf("%d", &x);

  /* Ako je broj negativan, koristi se njegova apsolutna vrednost
     i azurira se vrednost znaka broja. */
  if (x < 0) {
    x = abs(x);
    znak = -1;
  }

  /* Pozicija oznacava tezinu trenutne cifre rezultata i moze imati
     vrednosti 1, 10, 100, ... */
  pozicija = 1;
  rezultat = 0;
```

```
25   /* Ideja: u rezultatu se zadrzavaju cifre jedinice, stotine,.. Na
        primer, x=12345 Pre petlje: pozicija = 1, rezultat = 0 1.
27      iteracija: cifra = 5, rezultat = 0+5*1=5, x = 123, pozicija =
        10 2. iteracija: cifra = 3, rezultat = 5+3*10 = 35, x = 1,
29      pozicija = 100 3. iteracija: cifra = 1, rezultat = 35+1*100,
        x = 0, pozicija = 1000 Petlja se zavrsava jer x ima vrednost
31      0. */
   while (x > 0) {
33   /* Izdvajanje poslednje cifre. */
     cifra = x % 10;
35
     /* Rezultat se uvecava za vrednost cifre pomnozene vrednoscu
37      tezine njene pozicije u broju. */
     rezultat += cifra * pozicija;
39
     /* Uklanjanje poslednje dve cifre polaznog broja jer u rezultatu
41      treba da ostane svaka druga cifra. */
     x /= 100;
43
     /* Mnozenje pozicije sa 10, kako bi imala ispravnu vrednost u
45      sledecoj iteraciji. */
     pozicija *= 10;
47 }

49 /* Ispis rezultata */
   printf("Rezultat: %d\n", znak * rezultat);
51
   return 0;
53 }
```

Rešenje 1.5.26

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracije potrebnih promenljivih. */
5    int n, c1, c2, c3;
     int pozicija, rezultat;
7
     /* Ucitavanje vrednosti broja n. */
9    printf("Unesite broj: ");
     scanf("%d", &n);
11
     /* Provera ispravnosti ulaza. */
13   if (n <= 0) {
       printf("Greska: neispravan unos.\n");
15     return 1;
     }
17
     /* Ako broj nema bar tri cifre, rezultat ima vrednost unetog
19      broja. */
```

```
  if (n <= 99) {
21  printf("Rezultat: %d\n", n);
    return 0;
23 }

25 /* Izdvajanje poslednje tri cifre polaznog broja. */
   c1 = n % 10;
27 c2 = (n / 10) % 10;
   c3 = (n / 100) % 10;
29
   /* Poslednja cifra se uvek nalazi u rezultatu jer ona nema oba
31    suseda. Zato se rezultat inicijalizuje na poslednju cifru, a
      pozicija na 10. */
33 rezultat = c1;
   pozicija = 10;
35
   /* Petlja se izvrsava dok god broj ima bar tri cifre. */
37 while (n > 99) {
     /* Provera da li c2 treba da se nadje u rezultatu. Ako
39      treba, rezultat se uvecava za vrednost cifre pomnozene
        vrednoscu tezine njene pozicije u rezultatu i tezina
41      pozicije se mnozi sa 10. */
     if (c2 != c1 + c3) {
43     rezultat += c2 * pozicija;
       pozicija *= 10;
45   }

47   /* Vrsi se pomeranje na sledece tri cifre polaznog broja. Iz
        polaznog broja se brise poslednja cifra. Prva i druga cifra
49      su vec izracunate, samo se vrsi njihovo premestanje iz c2 i
        c3 u c1 i c2. Cifra c3 se racuna. */
51   n = n / 10;
     c1 = c2;
53   c2 = c3;
     c3 = (n / 100) % 10;
55 }

57 /* Po zavrsetku petlje, broj n je dvocifren i njegova cifra
      desetica odgovara vodecoj cifri polaznog broja. Vodeca cifra
59    polaznog broja uvek treba da se nadje u rezultatu jer nema oba
      suseda i iz tog razloga se dodaje na tekuci rezultat. */
61 rezultat += (n / 10) * pozicija;

63 /* Ispis rezultata. */
   printf("Rezultat: %d\n", rezultat);
65
   return 0;
67 }
```

Rešenje 1.5.27

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x, x_kopija, x_obrnuto;

  /* Ucitavanje vrednosti pocetnog broja. */
  printf("Unesite broj: ");
  scanf("%d", &x);

  /* Racunanje apsolutne vrednosti unetog broja. */
  if (x < 0)
    x = -x;

  /* Racunanje broja koji se dobije kada se broju x obrnu cifre. Na
     primer, od 12345 treba da se dobije 54321. Broj se obrce tako
     sto se u svakoj iteraciji njegova vrednost pomnozi sa 10 i
     doda mu se sledeca cifra polaznog broja.
     Za x_kopija=12345, x_obrnuto = 0
     1. iteracija: x_obrnuto = 0*10 + 5 = 5, x_kopija = 1234
     2. iteracija: x_obrnuto = 5*10 + 4 = 54, x_kopija = 123,
     3. iteracija: x_obrnuto = 54*10 + 3 = 543, x_kopija = 12,
     itd. */
  x_kopija = x;
  x_obrnuto = 0;
  while (x_kopija != 0) {
    x_obrnuto = x_obrnuto * 10 + x_kopija % 10;
    x_kopija /= 10;
  }

  /* Broj je palindrom ako je jednak broju koji se dobije
     obrtanjem njegovih cifara. Npr. x = 12321, x_obrnuto je
     takodje 12321. */
  if (x == x_obrnuto)
    printf("Broj je palindrom.\n");
  else
    printf("Broj nije palindrom.\n");

  return 0;
}
```

Rešenje 1.5.28

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  int fib1 = 0, fib2 = 1, fib3;
```

```
7
    /* Ucitavanje vrednosti broja n. */
9   printf("Unesite broj n: ");
    scanf("%d", &n);
11
    /* Provera ispravnosti ulaza. */
13  if (n < 0) {
      printf("Greska: neispravan unos.\n");
15    return 1;
    }
17
    /* Ako je n=0, F[0] = 0, slicno ako je n=1 F[1] = 1. */
19  if (n < 2) {
      printf("F[%d] = %d\n", n, n);
21    return 0;
    }
23
    fib3 = fib1 + fib2;
25  for (i = 2; i < n; i++) {
      /* Pomeranje na sledecu trojku. */
27    fib1 = fib2;
      fib2 = fib3;
29    fib3 = fib1 + fib2;
    }
31
    /* Ispis rezultata. */
33  printf("F[%d] = %d\n", n, fib3);

35  return 0;
}
```

Rešenje 1.5.29

```
#include <stdio.h>
2
int main() {
4   /* Deklaracija potrebne promenljive. */
    int a_n;
6
    /* Ucitavanje vrednosti prvog clana. */
8   printf("Unesite prvi clan:");
    scanf("%d", &a_n);
10
    /* Provera ispravnosti ulaza. */
12  if (a_n <= 0) {
      printf("Greska: neispravan unos.\n");
14    return 1;
    }
16
    /* Dok se ne dodje do clana koji je 1, stampa se vrednost
18     trenutnog clana i vrsi se izracunavanje narednog, po zadatoj
```

```
     formuli. */
  printf("Clanovi niza su:\n");
  while (a_n != 1) {
    printf("%d ", a_n);

    if (a_n % 2 != 0)
      a_n = (3 * a_n + 1) / 2;
    else
      a_n = a_n / 2;
  }

  /* Ispis jedinice na kraju. */
  printf("1\n");

  return 0;
}
```

Rešenje 1.5.30

```
#include <stdio.h>
#include <math.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int format, i;
  double sirina, duzina, nova_duzina;

  /* Ucitavanje formata papira. */
  printf("Unesite format papira: ");
  scanf("%d", &format);

  /* Provera ispravnosti ulaza. */
  if (format < 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* duzina/sirina = 1/sqrt(2)
     duzina*sirina = 1000mm x 1000mm =>

     duzina = sirina/sqrt(2)
     duzina*sirina = 1000mm x 1000mm =>

     sirina*sirina/sqrt(2) = 1000*1000
     sirina*sirina = sqrt(2) * 1000 * 1000
     sirina = sqrt(sqrt(2) * 1000 * 1000) =>

     duzina = sirina/sqrt(2) */
  sirina = sqrt(1000 * 1000 * sqrt(2));
  duzina = sirina / sqrt(2);
```

```
33   /* Racunanje duzine i sirine za uneti format. */
     for (i = 1; i <= format; i++) {
35     nova_duzina = sirina / 2;
       sirina = duzina;
37     duzina = nova_duzina;
     }
39
     /* Ispis rezultata. Napomena: Duzina i sirina su celi brojevi. */
41   printf("Dimenzije papira: %d %d\n", (int) duzina, (int) sirina);

43   return 0;
   }
```

Rešenje 1.5.31

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracija potrebne promenljive. */
5    char c;

7    /* I nacin ucitavanja: U samom uslovu petlje se vrsi ucitavanje
        jednog karaktera, njegovo smestanje u promenljivu c i provera
9       da li je ucitani karakter tacka. Zagrade oko (c=getchar()) su
        obavezne jer relacioni operator != ima veci prioritet od
11      dodele i kada ne bi postojale zagrade, redosled operacija bi
        bio: (c = (getchar() != '.')), sto znaci da bi se u c smestio
13      rezultat poredjenja, odnosno 0 ili 1. */
     while ((c = getchar()) != '.') {
15     /* Provera uslova i ispis odgovarajuceg karaktera. */
       if (c >= 'A' && c <= 'Z')
17       putchar(c + 'a' - 'A');
       else if (c >= 'a' && c <= 'z')
19       putchar(c - 'a' + 'A');
       else
21       putchar(c);
     }
23
     /* II nacin:
25      while(1) {
          c = getchar();
27        if(c == '.')
            break;
29
          if (c >= 'A' && c <= 'Z')
31          putchar(c + 'a' - 'A');
          else if (c >= 'a' && c <= 'z')
33          putchar(c - 'a' + 'A');
          else putchar(c);
35      } */
```

```
37   return 0;
}
```

Rešenje 1.5.32

```
#include <stdio.h>

int main() {
  /* Deklaracije i inicijalizacije potrebnih promenljivih. */
  char c;
  int broj_velikih = 0, broj_malih = 0;
  int broj_cifara = 0, suma_cifara = 0, broj_belina = 0;

  /* Petlja se zavrsava kada korisnik zada konstantu koja oznacava
     kraj ulaza (EOF konstantu). Ova konstanta se zadaje kombinacijom
     tastera CTRL+D i ima vrednost -1. */
  printf("Unesite tekst:\n");
  while ((c = getchar()) != EOF) {
    if (c >= 'A' && c <= 'Z')
      broj_velikih++;
    else if (c >= 'a' && c <= 'z')
      broj_malih++;
    else if (c >= '0' && c <= '9') {
      broj_cifara++;
      suma_cifara = suma_cifara + c - '0';
    } else if (c == '\t' || c == '\n' || c == ' ')
      broj_belina++;
  }

  /* Ispis rezultata. */
  printf("Velika slova: %d\nMala slova: %d\n", broj_velikih,
    broj_malih);
  printf("Cifre: %d\nBeline: %d\n", broj_cifara, broj_belina);
  printf("Suma cifara: %d\n", suma_cifara);

  return 0;
}
```

Rešenje 1.5.33

```
#include <stdio.h>

int main() {
  /* Deklaracije i inicijalizacije potrebnih promenljivih. */
  int n, i;
  int broj_a = 0, broj_e = 0, broj_i = 0, broj_o = 0, broj_u = 0;
  char c;

  /* Ucitavanje broja karaktera. */
  printf("Unesite broj n: ");
```

```
11    scanf("%d", &n);

13    /* Provera ispravnosti ulaza. */
      if (n < 0) {
15      printf("Greska: neispravan unos.\n");
        return 1;
17    }

19    /* Kako je korisnik nakon unosa broja n uneo oznaku za novi red,
         potrebno je preskociti taj novi red jer bi u suprotnom on bio
21       ucitan kao prvi od n karaktera (oznaka za novi red je
         regularan karakter kao sto je to 'a' ili ' '). */
23    getchar();

25    /* Ucitavanje karaktera i brojanje samoglasnika. */
      for (i = 0; i < n; i++) {
27      scanf("%c", &c);

29      switch (c) {
        case 'a':
31      case 'A':
          broj_a++;
33        break;
        case 'e':
35      case 'E':
          broj_e++;
37        break;
        case 'i':
39      case 'I':
          broj_i++;
41        break;
        case 'o':
43      case 'O':
          broj_o++;
45        break;
        case 'u':
47      case 'U':
          broj_u++;
49        break;
        }
51    }

53    /* Ispis rezultata. */
      printf("Samoglasnik a: %d\n", broj_a);
55    printf("Samoglasnik e: %d\n", broj_e);
      printf("Samoglasnik i: %d\n", broj_i);
57    printf("Samoglasnik o: %d\n", broj_o);
      printf("Samoglasnik u: %d\n", broj_u);
59
      return 0;
61  }
```

Rešenje 1.5.34

```
#include <stdio.h>

int main() {
  /* Deklaracije i inicijalizacije potrebnih promenljivih. */
  int n, i, broj_Z = 0, broj_i = 0, broj_m = 0, broj_a = 0;
  char novi_red, c;

  /* Ucitavanje broja karaktera. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ucitavanje karaktera. */
  for (i = 1; i <= n; i++) {
    printf("Unestite %d. karakter: ", i);

    /* Prvo se cita belina koja se nalazi nakon prethodnog unosa,
       pa tek posle procitane beline se cita uneti karakter. */
    scanf("%c%c", &novi_red, &c);

    switch (c) { /* Obrada ucitanog karaktera. */
    case 'Z':
      broj_Z++;
      break;
    case 'i':
      broj_i++;
      break;
    case 'm':
      broj_m++;
      break;
    case 'a':
      broj_a++;
      break;
    }
  }

  /* Ako su svi brojaci razliciti od nule, rec "Zima" se moze
     napisati pomocu unetih karaktera. */
  if (broj_Z && broj_i && broj_m && broj_a)
    printf("Moze se napisati rec Zima.\n");
  else
    printf("Ne moze se napisati rec Zima.\n");

  return 0;
}
```

Rešenje 1.5.35

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, i, suma_kubova;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n:");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje sume kubova svih brojeva iz intervala [1,n]. */
  suma_kubova = 0;
  for (i = 1; i <= n; i++)
    suma_kubova += i * i * i;

  /* Ispis rezultata. */
  printf("Suma kubova: %d\n", suma_kubova);

  return 0;
}
```

Rešenje 1.5.36 Pogledajte zadatak 1.5.35.

Rešenje 1.5.37

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, i;
  float x, suma, x_i;

  /* Ucitavanje vrednosti x i n. */
  printf("Unesite redom brojeve x i n: ");
  scanf("%f %d", &x, &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0 || x == 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Inicijalizacija vrednosti sume na 0, a vrednosti x^i na x. */
```

```
19    suma = 0;
      x_i = x;
21
      /* Promenljiva x^i ima vrednosti [x, x^2, ..., x^n]. Vrednost
23       sume se u svakoj iteraciji uvecava za i*x^i. */
      for (i = 1; i <= n; i++) {
25      suma += i * x_i;
        x_i *= x;
27    }

29    /* Ispis rezultata. */
      printf("S = %f\n", suma);
31
      return 0;
33  }
```

Rešenje 1.5.38 Pogledajte zadatak 1.5.37.

Rešenje 1.5.39

```
  #include <stdio.h>
2 #include <math.h>

4 int main() {
    /* Deklaracije potrebnih promenljivih. */
6   int i;
    float suma, clan, x, eps;
8
    /* Ucitavanje vrednosti x i eps. */
10  printf("Unesite x: ");
    scanf("%f", &x);
12
    printf("Unesite tacnost eps: ");
14  scanf("%f", &eps);

16  /* Inicijalizacija sume, prvog clana i brojaca. */
    suma = 0;
18  clan = 1;
    i = 1;
20
    /* U svakoj iteraciji na sumu se dodaje prethodno izracunati clan
22       sume i zatim se racuna sledeci clan. Petlja se prekida kada
         vrednost sledeceg clana postane manja ili jednaka eps. */
24  while (clan > eps) {
      suma += clan;
26    clan = clan * x / i;
      i++;
28  }

30  /* Ispis rezultata. */
    printf("S = %f\n", suma);
```

```
32
  return 0;
34 }
```

Rešenje 1.5.40

```
#include <stdio.h>
2 #include <math.h>

4 int main() {
  /* Deklaracije potrebnih promenljivih. */
6   int i;
  float suma, x, eps, clan;
8
  /* Ucitavanje vrednosti x i eps. */
10  printf("Unesite x: ");
  scanf("%f", &x);
12
  printf("Unesite tacnost eps: ");
14  scanf("%f", &eps);

16  /* Inicijalizacije. */
  suma = 0;
18  clan = 1;
  i = 1;
20
  /* Kako clanovi sume mogu biti negativni, potrebno je posmatrati
22     apsolutnu vrednost clana. */
  while (fabs(clan) > eps) {
24    suma += clan;

26    /* U svakoj iteraciji se racuna novi clan i mnozi se sa -1. Na
         taj nacin se postize da je vrednost clana naizmenicno
28       pozitivna i negativna. */
    clan = clan * x / i;
30    clan *= -1;

32    i++;
  }
34
  /* Ispis rezultata. */
36  printf("S = %f\n", suma);

38  return 0;
}
```

Rešenje 1.5.41

```
#include <stdio.h>
#include <math.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  double x, x_i, proizvod;

  /* Ucitavanje vrednosti x i n. */
  printf("Unesite redom brojeve x i n: ");
  scanf("%lf %d", &x, &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje trazenog proizvoda. */
  x_i = 1;
  proizvod = 1;
  for (i = 0; i < n; i++) {
    x_i *= x;
    proizvod *= 1 + cos(x_i);
  }

  /* Ispis rezultata. */
  printf("P = %lf\n", proizvod);

  return 0;
}
```

Rešenje 1.5.42

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  double razlomak;

  /* Ucitavanje vrednosti broja n.  */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
```

```
  }

  /* Razlomak se izracunava "od nazad", odnosno, krece se od
     najnizeg razlomka 1/n i od njega se nadalje formira sledeci,
     "visi" razlomak itd. Zavrsava se kada se stigne do koraka 0 +
     1/R. */
  razlomak = n;
  for (i = n - 1; i >= 0; i--)
    razlomak = i + 1 / razlomak;

  /* Ispis rezultata. */
  printf("R = %lf\n", razlomak);

  return 0;
}
```

Rešenje 1.5.43

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int i, n;
  float suma, x, clan;

  /* Ucitavanje vrednosti x i n. */
  printf("Unesite redom brojeve x i n: ");
  scanf("%f%d", &x, &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Inicijalizacije. */
  suma = 1;
  clan = 1;
  i = 2;

  /* Racunanje trazene sume. */
  while (i <= 2 * n) {
    /* Svaki clan sume se od prethodnog clana razlikuje za
       x^2/(i*(i-1)). */
    clan = clan * x * x / (i * (i - 1));
    clan *= -1;
    suma += clan;
    i += 2;
  }

  /* Ispis rezultata. */
```

```
  printf("S = %f\n", suma);

  return 0;
}
```

Rešenje 1.5.44

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  double clan, proizvod = 1;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje trazenog proizvoda. */
  clan = 1;
  for (i = 2; i <= n; i++) {
    clan = clan / i;
    proizvod *= 1 + clan;
  }

  /* Ispis rezultata. */
  printf("P = %lf\n", proizvod);

  return 0;
}
```

Rešenje 1.5.45

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  long int clan, suma = 0;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);

```

```
  /* Provera ispravnosti ulaza. */
  if (n < 5 || n % 2 == 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje trazene sume. */
  clan = -1 * 3;
  for (i = 5; i <= n; i += 2) {
    clan = -1 * clan * i;
    suma += clan;
  }

  /* Ispis rezultata. */
  printf("S = %ld\n", suma);

  return 0;
}
```

Rešenje 1.5.46

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i;
  double x, a, rezultat;

  /* Ucitavanje vrednosti ulaznih promenljivih. */
  printf("Unesite brojeve x i a: ");
  scanf("%lf%lf", &x, &a);
  printf("Unesite broj n: ");
  scanf("%d", &n);

  /* Provera ispravnosti ulaza. */
  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje vrednosti zadatog izraza. Krece se od
     rezultat = (x + a) ^ 2 i ide se ka spolja.
     Svaki put vrednost rezultata treba zameniti sa
     (rezultat + a) ^ 2. */
  rezultat = x;
  for (i = 0; i < n; i++)
    rezultat = (rezultat + a) * (rezultat + a);

  /* Ispis rezultata. */
  printf("Izraz = %lf\n", rezultat);

```

```
  return 0;
}
```

Rešenje 1.5.47

Rešenje (a)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Ispis tablice mnozenja dimenzije n*n. */
  for (i = 1; i <= n; i++) {
    for (j = 1; j <= n; j++) {
      /* Vrednost svakog polja je proizvod vrste i kolone. */
      printf("%3d ", i * j);
    }
    /* Na kraju svake vrste se ispisuje novi red. */
    printf("\n");
  }

  return 0;
}
```

Rešenje (b)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Brojac koji broji koliko brojeva je ispisano u jednom redu. */
  j = 0;
  for (i = 1; i <= n * n; i++) {
    printf("%3d ", i);

    j++;
    /* Kada je ispisano n brojeva u jednom redu, ispisuje se znak
```

```
18        za novi red, da bi ispis krenuo u novom redu i vrednost
          brojaca j se postavlja na 0 jer u novom redu jos ni jedan
20        broj nije ispisan. */
      if (j == n) {
22      j = 0;
        printf("\n");
24    }
  }
26
  return 0;
28}
```

Rešenje (c)

```
#include <stdio.h>
2
int main() {
4  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;
6
  /* Ucitavanje vrednosti broja n. */
8  printf("Unesite broj n: ");
  scanf("%u", &n);
10
  /* Ispis trazene tablice. */
12  for (i = 1; i <= n; i++) {
    for (j = 0; j < n; j++)
14      if ((j + i) % n == 0)
        printf("%3d", n);
16      else
        printf("%3d", (j + i) % n);
18
    printf("\n");
20  }

22  return 0;
}
```

Rešenje (d)

```
1#include <stdio.h>

3int main() {
  /* Deklaracija potrebnih promenljivih. */
5  unsigned int n, i, j;

7  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
9  scanf("%u", &n);
```

```
11    /* Ispis trazenog trougla. */
      for (i = 0; i < n; i++) {
13      for (j = 0; j < n - i; j++)
          printf("(%d, %d)", i, j);
15
        printf("\n");
17    }

19    return 0;
  }
```

Rešenje 1.5.48

Rešenje (a)

```
  #include <stdio.h>
2
  int main() {
4   /* Deklaracija potrebnih promenljivih. */
    unsigned int n, i, j;
6
    /* Ucitavanje vrednosti broja n. */
8   printf("Unesite broj n: ");
    scanf("%u", &n);
10
    /* Kvadrat predstavlja tabelu sa n vrsta gde svaka vrsta sadrzi n
12     polja i u svakom polju je upisana zvezdica. */
    for (i = 0; i < n; i++) {
14    for (j = 0; j < n; j++)
        printf("*");
16    printf("\n");
    }
18
    return 0;
20 }
```

Rešenje (b)

```
  #include <stdio.h>
2
  int main() {
4   /* Deklaracija potrebnih promenljivih. */
    unsigned int n, i, j;
6
    /* Ucitavanje vrednosti broja n. */
8   printf("Unesite broj n: ");
    scanf("%u", &n);
10
```

```
  /* Kvadrat predstavlja tabelu sa n vrsta i n kolona u kojoj se
     na ivicama nalaze zvezdice, a u unutrasnjosti praznine. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      /* Provera da li je u pitanju ivica. */
      if (j == 0 || j == n - 1 || i == 0 || i == n - 1)
        printf("*");
      else
        printf(" ");
    }
    printf("\n");
  }

  return 0;
}
```

Rešenje (c)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Kvadrat predstavlja tabelu sa n vrsta i n kolona u kojoj se
     na ivicama i glavnoj dijagonali nalaze zvezdice, a na ostalim
     mestima praznine. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      /* Provera da li je ivica ili glavna dijagonala. */
      if (j == 0 || j == n - 1 || i == 0 || i == n - 1 || i == j)
        printf("*");
      else
        printf(" ");
    }
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.49

```
#include <stdio.h>

int main() {
```

```
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Veliko slovo X se dobija tako sto se na dijagonalama kvadrata
     ispisuju zvezdice, a na ostalim mestima blanko. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      /* Provera da li je mesto glavne ili sporedne dijagonale. */
      if (i == j || i + j == n - 1)
        printf("*");
      else
        printf(" ");
    }
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.50

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje znaka plus ispisom karaktera '+' na sredisnjoj
     vrsti i koloni, a blanko karaktera na ostalim mestima. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
      if (i == n / 2 || j == n / 2)
        printf("+");
      else
        printf(" ");
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.51

Rešenje (a)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje trazenog trougla. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n - i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (b)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje trazenog trougla. */
  for (i = 0; i < n; i++) {
    for (j = 0; j <= i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (c)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje trazenog trougla. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < i; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < n - i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (d)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje trazenog trougla. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j <= i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (e)

```
#include <stdio.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  unsigned int n, i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Iscrtavanje gornjeg dela trazenog trougla. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j <= i; j++)
      printf("*");
    printf("\n");
  }

  /* Iscrtavanje donjeg dela trazenog trougla. */
  for (i = 1; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < i; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < n - i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (f)

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n, i, j;
  char c, novi_red;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);
```

```
12    /* Ucitavanje karaktera koji ce se koristiti za iscrtavanje.
         Napomena: Voditi racuna da treba preskociti novi red koji
14       korisnik zadaje nakon unosa broja n. */
      printf("Unesite karakter c: ");
16    scanf("%c%c", &novi_red, &c);

18    /* Iscrtavanje trazenog trougla. Iscrtavaju se samo ivice
         trougla, ostalo se popunjava belinama. */
20    for (i = 0; i < n; i++) {
        for (j = 0; j <= i; j++)
22        if (i == n - 1 || j == 0 || j == i)
            printf("%c", c);
24        else
            printf(" ");
26      printf("\n");
      }
28
      return 0;
30  }
```

Rešenje 1.5.52

Rešenje (a)

```
1   #include <stdio.h>

3   int main() {
      /* Deklaracija potrebnih promenljivih. */
5     unsigned int n, i, j;

7     /* Ucitavanje vrednosti broja n. */
      printf("Unesite broj n: ");
9     scanf("%u", &n);

11    /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
      for (i = 0; i < n; i++) {
13      /* Ispis belina koje prethode zvezdicama. */
        for (j = 0; j < n - i - 1; j++)
15        printf(" ");
        /* Ispis potrebnog broja zvezdica. */
17      for (j = 0; j < 2 * i + 1; j++)
          printf("*");
19      printf("\n");
      }
21
      return 0;
23  }
```

Rešenje (b)

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Brojac i odredjuje koliko redova se ispisuje. Radi lakseg
     izracunavanja koliko zvezdica i praznina je potrebno ispisati
     u svakom redu, i se postavlja na n-1 i smanjuje u svakoj
     iteraciji petlje. */
  for (i = n - 1; i >= 0; i--) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < 2 * i + 1; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (c)

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Slika se crta iz dva dela. */

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
```

```
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < 2 * i + 1; j++)
      printf("*");
    printf("\n");
  }

  /* II deo: crtanje donjeg trougla. Kako je prvi red donjeg
     trougla vec ispisan (poslednji red gornjeg trougla), potrebno
     je naciniti jednu iteraciju manje. */
  for (i = n - 2; i >= 0; i--) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < 2 * i + 1; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje (d)

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Posle belina se ispisuje sam trougao. Ako je brojac na ivici
       onda se ispisuje zvezdica, a inace praznina. Takodje,
       proverava se da li se ispisuje poslednji red (i==n) i u njemu
       se ispisuje svaki drugi put zvezdica, a inace praznina. */
    for (j = 0; j < 2 * i + 1; j++)
      if (j == 0 || j == 2 * i || (i == n - 1 && j % 2 == 0))
        printf("*");
      else
        printf(" ");
    printf("\n");
  }
```

```
29   return 0;
}
```

Rešenje (c)

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 0; i < n; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis trougla. */
    for (j = 0; j < 2 * i + 1; j++)
      if (j == 0 || j == 2 * i || (i == n - 1 && j % 2 == 0))
        printf("*");
      else
        printf(" ");
    printf("\n");
  }

  /* II deo: crtanje donjeg trougla. Kako je prvi red donjeg
     trougla vec ispisan (poslednji red gornjeg trougla), potrebno
     je naciniti jednu iteraciju manje. */
  for (i = n - 2; i >= 0; i--) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
    /* Ispis potrebnog broja zvezdica. */
    for (j = 0; j < 2 * i + 1; j++)
      if (j == 0 || j == 2 * i)
        printf("*");
      else
        printf(" ");
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.53

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Strelica se moze posmatrati kao spojena dva pravougla trougla
     kojima se ispisuje hipotenuza i jedna kateta. */

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 0; i < n; i++) {
    for (j = 0; j <= i; j++)
      /* Provera da li se ispisuje karakter na hipotenuzi (j == i)
         ili da se ispisuje poslednji red (i == n-1). */
      if (j == i || i == n - 1)
        printf("*");
      else
        printf(" ");
    printf("\n");
  }

  /* II deo: crtanje donjeg dela slike, odnosno donji trougao.
     Brojac i odredjuje koji red donjeg trougla se trentno
     iscrtava. Kako je prvi red donjeg trougla vec iscrtan (to je
     poslednji red gornjeg trougla), brojac se postavlja na 1. */
  for (i = 1; i < n; i++) {
    for (j = 0; j < n - i; j++)
      /* Provera da li se ispisuje hipotenuza. */
      if (j == n - i - 1)
        printf("*");
      else
        printf(" ");
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.54

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
```

```
  unsigned int n;
  int i, j, k;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Brojac j odredjuje koliko se karaktera (praznina i zvezdica)
     ispisuje u svakom redu.
     U svakom drugom redu ovaj broj se povecava za 2.
     Na pocetku je 1 (jer se ispisuje samo jedna zvezdica). */
  j = 1;

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 1; i <= n; i++) {
    /* U svakom drugom redu broj karaktera koji treba da se ispisu
       se uvecava za 2. */
    if (i % 2 == 0)
      j += 2;

    /* Ispisuje se j karaktera. */
    for (k = 0; k < j; k++)
      /* U svakom parnom redu se naiazmenicno ispisuju zvezdica i
         praznina. */
      if (i % 2 == 0) {
        if (k % 2 == 0)
          printf("*");
        else
          printf(" ");
      } else {
        /* U svakom neparnom redu se ispisuju samo zvezdice. */
        printf("*");
      }
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.55

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n, m;
  int i, j;

  /* Ucitavanje dimenzije slike. */
  printf("Unesite brojeve n i m: ");
  scanf("%u%u", &n, &m);
```

```
  /* Brojac i odredjuje koji red slike se trenutno ispisuje. Ukupno
     ima m redova. */
  for (i = 1; i <= m; i++) {
    /* Brojac j oznacava koja kolona se trenutno ispisuje. Za
       svaki kvadrat se racuna duzina bez poslednje ivice. Kvadrat
       je sastavljen od (m-1) zvezdice i (m-1) praznine (praznine
       se nalaze izmedju zvezdica). Znaci, ukupna duzina je 2*(m-1)
       karakter, a kako ima n kvadrata plus jedna kolona za
       krajnje desnu ivicu, duzina je n*2*(m-1) + 1. */
    for (j = 0; j <= n * 2 * (m - 1); j++)
      /* Provera da li se ispisuje prvi ili poslednji red. */
      if (i == 1 || i == m)
        /* Naizmenican ispis zvezdice i praznine. */
        if (j % 2 == 0)
          printf("*");
        else
          printf(" ");
      else
        /* Na ivicama kvardata se iscrtavaju zvezdice, a na ostalim
           mestima beline. */
      if (j % (2 * (m - 1)) == 0)
        printf("*");
      else
        printf(" ");

    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.56

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Romb se crta crtanjem dva spojena trougla koji se nezavisno
     iscrtavaju. */

  /* Brojac i odredjuje koji red slike se trenutno ispisuje. */
  for (i = 0; i < n; i++) {
    /* Ispis zvezdica koje prethode karakterima -. */
```

```
    for (j = 0; j < n - i; j++)
      printf("*");
    /* Ispis karaktera -. */
    for (j = 0; j < 2 * i; j++)
      printf("-");
    /* Ispis zvezdica koje su nakon karaktera -. */
    for (j = 0; j < n - i; j++)
      printf("*");
    printf("\n");
  }

  /* II deo: crtanje donjeg trougla. Kako je prvi red donjeg
     trougla vec ispisan (poslednji red gornjeg trougla), potrebno
     je naciniti jednu iteraciju manje. */
  for (i = n - 2; i >= 0; i--) {
    /* Ispis zvezdica koje prethode karakterima -. */
    for (j = 0; j < n - i; j++)
      printf("*");
    /* Ispis karaktera -. */
    for (j = 0; j < 2 * i; j++)
      printf("-");
    /* Ispis zvezdica koje su nakon karaktera -. */
    for (j = 0; j < n - i; j++)
      printf("*");
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.57

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Slika se sastoji iz dva dela, trougla i kvadrata i svaki deo
     se nezavisno iscrtava. */

  /* I deo: crtanje trougla (krova). */
  for (i = 0; i < n - 1; i++) {
    /* Ispis belina koje prethode zvezdicama. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");
```

```
    /* Ispis trougla. */
    for (j = 0; j < 2 * i + 1; j++)
      if (j == 0 || j == 2 * i)
        printf("*");
      else
        printf(" ");
    printf("\n");
  }

  /* II deo: crtanje kvadrata. Da bi iscrtavanje bilo lakse
     istovremeno se ispisuju dva karaktera. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
      /* Provera da li je ivica. */
      if (j == 0 || j == n - 1 || i == 0 || i == n - 1)
        printf("* ");
      else
        printf("  ");
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5.58

```
#include <stdio.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int n;
  int i, j;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%u", &n);

  /* Prva petlja oznacava broj 'serija' koje ce se ispisati. Na
     primer, za n=5, prva serija je 1 2 3 4 5, druga serija je 2 3
     4 i treca serija je 3. Kako se u svakoj sledecoj seriji broj
     brojeva smanjuje za 2, do 0 karaktera u seriji se dolazi posle
     n/2 koraka, ali zaokruzeno navise (5/2 = 2.5 --> 3), a to je
     isto sto i celobrojno (n+1)/2. */
  for (i = 1; i <= (n + 1) / 2; i++) {
    /* U svakoj seriji se ispisuju brojevi izmedju i i n-i+1. */
    for (j = i; j <= n + 1 - i; j++)
      printf("%d ", j);
  }

  /* II nacin:
```

```
25   int levo = 1, desno = n-1;
     while (levo <= desno) {
27     // Ispis jedne serije.
       for (j = levo; j <= desno; j++)
29       printf(" %d", j);

31     // Pomeranje leve i desne granice.
       levo++;
33     desno--;
     } */
35
     return 0;
37 }
```

Rešenje 1.5.59

```
1  #include <stdio.h>

3  int main() {
     /* Deklaracije potrebnih promenljivih. */
5    unsigned int n;
     int i, j;
7
     /* Ucitavanje vrednosti broja n. */
9    printf("Unesite broj n: ");
     scanf("%u", &n);
11
     /* Brojac i je redni broj vrste koja se ispisuje. */
13   for (i = 1; i <= n; i++) {
       /* Ispis brojeva izmedju 1 i n, sa korakom i. */
15     for (j = 1; j <= n; j += i)
         printf("%d ", j);
17
       printf("\n");
19   }

21   return 0;
   }
```

1.7 Funkcije

Zadatak 1.7.1 Napisati funkciju `int min(int x, int y, int z)` koja izračunava minimum tri broja. Napisati program koji učitava tri cela broja i ispisuje njihov minimum.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve: 19 8 14
 Minimum: 8
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve: -6 11 -12
 Minimum: -12
```

Zadatak 1.7.2 Napisati funkciju `float razlomljeni_deo(float x)` koja izračunava razlomljeni deo broja x. Napisati program koji učitava jedan realan broj i ispisuje njegov razlomljeni deo na šest decimala.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 8.235
 Razlomljeni deo: 0.235000
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: -5.11
 Razlomljeni deo: 0.110000
```

Zadatak 1.7.3 Napisati funkciju `int zbir_delilaca(int n)` koja izračunava zbir delilaca broja n. Napisati program koji učitava ceo pozitivan broj k i ispisuje zbir delilaca svakog broja od 1 do k. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj k: 6
 1 3 4 7 6 12
```

Primer 2

```
Interakcija sa programom:
 Unesite broj k: -2
 Greska: neispravan unos.
```

Zadatak 1.7.4 Napisati funkciju `int je_stepen(unsigned x, unsigned n)` koja za dva broja x i n utvrđuje da li je x neki stepen broja n. Ukoliko jeste, funkcija vraća izložilac stepena, a u suprotnom vraća -1. Napisati program koji učitava dva neoznačena broja i ispisuje da li vrednost prvog broja odgovara vrednosti nekog stepena drugog broja (i kog).

Primer 1

```
Interakcija sa programom:
 Unesite dva broja: 81 3
 Jeste: 81 = 3^4
```

Primer 2

```
Interakcija sa programom:
 Unesite dva broja: 162 11
 Broj 162 nije stepen broja 11.
```

Zadatak 1.7.5 Napisati funkciju `int euklid(int x, int y)` koja za dva data cela broja određuje najveći zajednički delilac primenom Euklidovog algoritma. Napisati program koji učitava dva cela broja i ispisuje vrednost njihovog najvećeg zajedničkog delioca.

Primer 1

```
Interakcija sa programom:
 Unesite dva cela broja: 1024 832
 Najveci zajednicki delilac: 64
```

Primer 2

```
Interakcija sa programom:
 Unesite dva cela broja: -900 112
 Najveci zajednicki delilac: 4
```

Zadatak 1.7.6 Napisati funkciju `float zbir_reciprocnih(int n)` koja za dato n vraća zbir recipročnih vrednosti brojeva od 1 do n. Napisati program koji učitava ceo pozitivan broj n i ispisuje odgovarajući zbir zaokružen na dve decimale. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 10
 Zbir reciprocnih: 2.93
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 100
 Zbir reciprocnih: 5.19
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -100
 Greska: neispravan unos.
```

Zadatak 1.7.7 Napisati funkciju `int prebrojavanje(float x)` koja prebrojava koliko puta se broj x pojavljuje u nizu brojeva koji se unose sve do unosa broja nula. Napisati program koji učitava vrednost broja x i ispisuje koliko puta se njegova vrednost pojavila u unetom nizu.

Primer 1

```
Interakcija sa programom:
 Unesite broj x: 2.84
 Unesite brojeve:
 8.13 2.84 5 21.6 2.84 11.5 0
 Broj pojavljivanja broja 2.84: 2
```

Primer 2

```
Interakcija sa programom:
 Unesite broj x: -1.17
 Unesite brojeve:
 -128.35 8.965 8.968 89.36 0
 Broj pojavljivanja broja -1.17: 0
```

Zadatak 1.7.8 Broj je prost ako je deljiv samo sa 1 i samim sobom.

(a) Napisati funkciju `int prost(int x)` koja ispituje da li je dati ceo broj prost. Funkcija treba da vrati jedinicu ako je broj prost ili nulu u suprotnom.

(b) Napisati funkciju `void prvih_n_prostih(int n)` koja ispisuje prvih n prostih brojeva.

(c) Napisati funkciju `void prosti_brojevi_manji_od_n(int n)` koja ispisuje sve proste brojeve manje od broja n.

Napisati program koji učitava pozitivan ceo broj n i ispisuje prvih n prostih brojeva, kao i sve proste brojeve manje od n. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
 Prvih n prostih: 2 3 5 7 11
 Prosti manji od n: 2 3
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 2
 Prvih n prostih: 2 3
 Prosti manji od n: ne postoje
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -11
 Greska: neispravan unos.
```

Zadatak 1.7.9 Rešiti sledeće zadatke korišćenjem funkcija.

(a) Zadatak 1.1.2 rešiti korišćenjem funkcija `int kvadrat(int x)` koja računa kvadrat datog broja i `int kub(int x)` koja računa kub datog broja.

(b) Zadatak 1.3.2 rešiti korišćenjem funkcije `float apsolutna_vrednost(float x)` koja izračunava apsolutnu vrednost datog broja.

(c) Zadatak 1.5.7 rešiti korišćenjem funkcije `float stepen(float x, int n)` koja računa vrednost n-tog stepena realnog broja x.

(d) Zadatak 1.5.28 rešiti korišćenjem funkcije `int fibonaci(int n)` koja računa n-ti element Fibonačijevog niza.

Zadatak 1.7.10 Napisati funkciju `float aritmeticka_sredina(int n)` koja računa aritmetičku sredinu cifara datog broja. Napisati i program koji učitava ceo broj i ispisuje aritmetičku sredinu njegovih cifara zaokruženu na tri decimale.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 461
 Aritmeticka sredina: 3.667
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 1001
 Aritmeticka sredina: 0.500
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -84723
 Aritmeticka sredina: 4.800
```

Zadatak 1.7.11 Napisati funkciju `int sadrzi(int x, int c)` koja ispituje da li se cifra c nalazi u zapisu celog broja x. Funkcija treba da vrati jedinicu ako se cifra nalazi u broju, a nulu inače. Napisati program koji učitava jedan ceo broj i jednu cifru i u zavisnosti od toga da li se uneta cifra nalazi u zapisu unetog broja, ispisuje odgovarajuću poruku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj i cifru: 17890 7
 Cifra 7 se nalazi u zapisu broja 17890.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj i cifru: 19 6
 Cifra 6 se ne nalazi u zapisu broja 19.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj i cifru: 17890 26
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj i cifru: -1982 9
 Cifra 9 se nalazi u zapisu broja -1982.
```

Zadatak 1.7.12 Napisati funkciju `int broj_neparnih_cifara(int x)` koja određuje broj neparnih cifara u zapisu datog celog broja. Napisati program koji učitava cele brojeve sve do unosa broja nula i ispisuje broj neparnih cifara svakog unetog broja.

Primer 1

```
Interakcija sa programom:
 Unesite cele brojeve:
 2341
 Broj neparnih cifara: 2
 78
 Broj neparnih cifara: 1
 800
 Broj neparnih cifara: 0
 -99761
 Broj neparnih cifara: 4
 0
```

Primer 2

```
Interakcija sa programom:
 Unesite cele brojeve:
 987611
 Broj neparnih cifara: 4
 135
 Broj neparnih cifara: 3
 -701
 Broj neparnih cifara: 2
 602
 Broj neparnih cifara: 0
 -884
 Broj neparnih cifara: 0
 79901
 Broj neparnih cifara: 4
 0
```

Zadatak 1.7.13 Napisati program za ispitivanje svojstava cifara datog celog broja.

(a) Napisati funkciju `int sve_parne_cifre(int x)` koja ispituje da li se dati ceo broj sastoji isključivo iz parnih cifara. Funkcija treba da vrati jedinicu ako su sve cifre broja parne, a nulu inače.

(b) Napisati funkciju `int sve_cifre_jednake(int x)` koja ispituje da li su sve cifre datog celog broja jednake. Funkcija treba da vrati jedinicu ako su sve cifre broja jednake, a nulu inače.

Program učitava ceo broj i u zavisnosti od toga da li su navedena svojstva ispunjena ili ne, ispisuje odgovarajuću poruku.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 86422
 Sve cifre broja su parne.
 Cifre broja nisu jednake.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 55555
 Broj sadrzi bar jednu neparnu cifru.
 Cifre broja su jednake.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: -88
 Sve cifre broja su parne.
 Cifre broja su jednake.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj i cifru: -342
 Broj sadrzi bar jednu neparnu cifru.
 Cifre broja nisu jednake.
```

Zadatak 1.7.14 Napisati funkciju `int ukloni(int n, int p)` koja menja broj n tako što iz njegovog zapisa uklanja cifru na poziciji p. Pozicije se broje sa desna na levo. Cifra jedinica ima poziciju 1. Napisati program koji učitava redni broj pozicije i zatim za cele brojeve koji se unose sve do unosa broja nula, ispisuje brojeve kojima je uklonjena cifra na poziciji p. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite poziciju: 3
 Unesite broj: 1210
 Novi broj: 110
 Unesite broj: 18
 Novi broj: 18
 Unesite broj: 3856
 Novi broj: 356
 Unesite broj: 0
```

Primer 2

```
Interakcija sa programom:
 Unesite poziciju: 1
 Unesite broj: -9632
 Novi broj: -963
 Unesite broj: -2
 Novi broj: 0
 Unesite broj: 246
 Novi broj: 24
 Unesite broj: -52
 Novi broj: -5
 Unesite broj: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite poziciju: 0
 Greska: neispravan unos.
```

Zadatak 1.7.15 Napisati funkciju `int zapis(int x, int y)` koja proverava da li se brojevi x i y zapisuju pomoću istih cifara. Funkcija treba da vrati jedinicu ako je uslov ispunjen, a nulu inače. Napisati program koji učitava dva cela broja i ispisuje da li je za njih pomenuti uslov ispunjen ili ne.

Primer 1

```
Interakcija sa programom:
 Unesite dva broja: 251 125
 Uslov je ispunjen.
```

Primer 2

```
Interakcija sa programom:
 Unesite dva broja: 8898 9988
 Uslov nije ispunjen.
```

Primer 3

```
Interakcija sa programom:
 Unesite dva broja: -7391 1397
 Uslov je ispunjen.
```

Primer 4

```
Interakcija sa programom:
 Unesite dva broja: -777 77
 Uslov nije ispunjen.
```

Zadatak 1.7.16 Napisati funkciju `int neopadajuce(int n)` koja ispituje da li su cifre datog celog broja u neopadajućem poretku. Funkcija treba da vrati jedinicu ako cifre ispunjavaju uslov, a nulu inače. Napisati program koji učitava ceo broj i ispisuje poruku da li su cifre unetog broja u neopadajućem poretku.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 2289
 Cifre su u neopadajucem poretku.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 5
 Cifre su u neopadajucem poretku.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 6628
 Cifre nisu u neopadajucem poretku.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj: -23
 Cifre su u neopadajucem poretku.
```

Zadatak 1.7.17 Napisati funkciju `int par_nepar(int n)` koja ispituje da li su cifre datog celog broja naizmenično parne i neparne. Funkcija treba da vrati jedinicu ako cifre ispunjavaju uslov, a nulu inače. Napisati program koji učitava ceo broj i ispisuje da li on ispunjava pomenuti uslov ili ne.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 2749
 Broj ispunjava uslov.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: -963
 Broj ispunjava uslov.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: 27449
 Broj ne ispunjava uslov.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj n: 5
 Broj ispunjava uslov.
```

Zadatak 1.7.18 Napisati funkciju `int rotacija(int n)` koja rotira cifre zadatog celog broja za jednu poziciju u levo. Napisati program koji za brojeve koji se unose sve do unosa broja nula ispisuje odgovarajuće rotirane brojeve.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 146
 Novi broj: 461
 Unesite broj: 18
 Novi broj: 81
 Unesite broj: 3856
 Novi broj: 8563
 Unesite broj: 7
 Novi broj: 7
 Unesite broj: 0
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 89
 Novi broj: 98
 Unesite broj: -369
 Novi broj: -693
 Unesite broj: -55281
 Novi broj: -52815
 Unesite broj: 0
```

Zadatak 1.7.19 Za dati broj može se formirati niz tako da je svaki sledeći član niza dobijen kao suma cifara prethodnog člana niza. Broj je *srećan* ako

se dati niz završava jedinicom. Napisati funkciju `int srecan(int x)` koja vraća jedinicu ako je broj srećan, a nulu inače. Napisati program koji za uneti pozitivan ceo broj n ispisuje sve srećne brojeve od 1 do n. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 100
 Srecni brojevi:
 1 10 19 28 37 46 55 64 73 82 91 100
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 0
 Greska: neispravan unos.
```

Zadatak 1.7.20 Prirodan broj a je Armstrongov ako je jednak sumi n-tih stepena svojih cifara, pri čemu je n broj cifara broja a. Napisati funkciju `int armstrong(int x)` koja vraća jedinicu ako je broj Armstrongov, a nulu inače. Napisati program koji za učitani pozitivan ceo broj proverava da li je Armstrongov. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj: 1634
 Broj je Armstrongov.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj: 118
 Broj nije Armstrongov.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj: 0
 Greska: neispravan unos.
```

Zadatak 1.7.21 Napisati funkciju `double e_na_x(double x, double eps)` koja računa vrednost e^x kao parcijalnu sumu reda $\sum_{n=0}^{\infty} \frac{x^n}{n!}$, pri čemu se sumiranje sprovodi sve dok je sabirak po apsolutnoj vrednosti veći od date tačnosti `eps`. Napisati program koji učitava dva realna broja x i *eps* i ispisuje izračunatu vrednost e^x.

Primer 1

```
Interakcija sa programom:
 Unesite broj x: 5
 Unesite eps: 0.001
 Rezultat: 148.412951
```

Primer 2

```
Interakcija sa programom:
 Unesite broj x: -3
 Unesite eps: 0.0001
 Rezultat: 0.049796
```

Zadatak 1.7.22 Napisati funkciju `void ispis(float x, float y, int n)` koja za dva realna broja x i y i jedan pozitivan ceo broj n ispisuje vrednosti sinusne funkcije u n ravnomerno raspoređenih tačaka intervala $[x, y]$. Napisati program koji učitava granice intervala i broj tačaka i ispisuje odgovarajuće vrednosti sinusne funkcije zaokružene na četiri decimale. U slučaju neispravnog unosa,

ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dva realna broja: 7 31
 Unesite broj n: 6
 Rezultat:
 sin(7.0000) = 0.6570
 sin(11.8000) = -0.6935
 sin(16.6000) = -0.7784
 sin(21.4000) = 0.5573
 sin(26.2000) = 0.8759
 sin(31.0000) = -0.4040
```

Primer 2

```
Interakcija sa programom:
 Unesite dva realna broja: -8.32 20.5
 Unesite broj n: 5
 Rezultat:
 sin(-8.3200) = -0.8934
 sin(-1.1150) = -0.8979
 sin(6.0900) = -0.1920
 sin(13.2950) = 0.6658
 sin(20.5000) = 0.9968
```

Primer 3

```
Interakcija sa programom:
 Unesite dva realna broja: 8 8
 Greska: neispravan unos.
```

Primer 1

```
Interakcija sa programom:
 Unesite dva realna broja: 7 32
 Unesite broj n: -10
 Greska: neispravan unos.
```

Zadatak 1.7.23 Napisati funkciju `char sifra(char c, int k)` koja za dati karakter c određuje šifru na sledeći način: ukoliko je c slovo, šifra je slovo koji se nalazi k pozicija pre njega u engelskoj abecedi. Karakteri koji nisu slova se ne šifruju. Šifrovanje treba da bude kružno, što znači da je, na primer, šifra za karakter b i pomeraj 2 karakter z. Napisati program koji učitava nenegativan ceo broj k, a zatim i karaktere sve do kraja ulaza i nakon svakog učitanog karaktera ispisuje njegovu šifru. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj k: 2
 Unesite tekst (CTRL+D za prekid):
 U svetu postoji jedno carstvo
 S qtcrs nmqrmhg hcblm aypqrtm
 U njemu caruje drugarstvo.
 S lhcks aypshc bpseypqrtm.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj k: -2
 Greska: neispravan unos.
```

Zadatak 1.7.24 Rešiti sledeće zadatke korišćenjem funkcija.

(a) Zadatak 1.5.31 rešiti korišćenjem funkcije `char konverzija(char c)` koja malo slovo pretvara u odgovarajuće veliko i obrnuto.

(b) Zadatak 1.5.32 rešiti korišćenjem funkcije `void prebrojavanje()` koja učitava karaktere sve do kraja ulaza i ispisuje broj malih slova, velikih slova, cifara, belina, kao i sumu svih unetih cifara.

Zadatak 1.7.25 Napisati program koji učitava tri cela broja koja predstavljaju dan, mesec i godinu i ispisuje datum sledećeg dana. Zadatak rešiti korišćenjem narednih funkcija.

(a) `int prestupna(int godina)` koja za zadatu godinu proverava da li je prestupna. Funkcija treba da vrati jedinicu ako je godina prestupna ili nulu ako nije.

(b) `int broj_dana(int mesec, int godina)` koja za dati mesec i godinu vraća broj dana u datom mesecu.

(c) `int ispravan(int dan, int mesec, int godina)` koja za dati datum proverava da li je ispravan.

(d) `void sledeci_dan(int dan, int mesec, int godina)` koja za dati datum ispisuje datum sledećeg dana.

U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite datum: 24.8.1998.
 Datum sledeceg dana je: 25.8.1998.
```

Primer 2

```
Interakcija sa programom:
 Unesite datum: 31.12.1789.
 Datum sledeceg dana je: 1.1.1790.
```

Primer 3

```
Interakcija sa programom:
 Unesite datum: 28.2.2003.
 Datum sledeceg dana je: 1.3.2004.
```

Primer 4

```
Interakcija sa programom:
 Unesite datum: 31.4.2004.
 Greska: neispravan unos.
```

Zadatak 1.7.26 Napisati funkciju `int od_nove_godine(int dan, int mesec, int godina)` koja određuje koliko je dana proteklo od Nove godine do datog datuma. Napisati program koji učitava tri cela broja koja predstavljaju dan, mesec i godinu i ispisuje koliko dana je proteklo od Nove godine. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite datum: 24.8.1998.
 Broj dana od Nove godine je: 235
```

Primer 2

```
Interakcija sa programom:
 Unesite datum: 31.12.1680.
 Broj dana od Nove godine je: 366
```

Primer 3

```
Interakcija sa programom:
 Unesite datum: 28.2.2003.
 Broj dana od Nove godine je: 58
```

Primer 4

```
Interakcija sa programom:
 Unesite datum: 31.4.2004.
 Greska: neispravan unos.
```

Zadatak 1.7.27 Napisati funkciju `int do_kraja_godine(int dan, int mesec, int godina)` koja određuje broj dana od datog datuma do kraja godine. Napisati program koji učitava tri cela broja koja predstavljaju dan, mesec i godinu i ispisuje broj dana do kraja godine. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite datum: 24.8.1998.
 Broj dana do Nove godine je: 129
```

Primer 2

```
Interakcija sa programom:
 Unesite datum: 31.12.1680.
 Broj dana do Nove godine je: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite datum: 28.2.2004.
 Broj dana do Nove godine je: 307
```

Primer 4

```
Interakcija sa programom:
 Unesite datum: 31.4.2004.
 Greska: neispravan unos.
```

Zadatak 1.7.28 Napisati funkciju `int broj_dana_izmedju(int dan1, int mesec1, int godina1, int dan2, int mesec2, int godina2)` koja određuje broj dana između dva datuma. Napisati program koji učitava dva datuma u formatu `dd.mm.gggg` i na standarni izlaz ispisuje broj dana između ta dva datuma. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite prvi datum: 12.3.2008.
 Unesite drugi datum: 5.12.2008.
 Broj dana izmedju dva datuma je: 268
```

Primer 2

```
Interakcija sa programom:
 Unesite prvi datum: 26.9.1986.
 Unesite drugi datum: 2.2.1701.
 Broj dana izmedju dva datuma je: 104301
```

Primer 3

```
Interakcija sa programom:
 Unesite prvi datum: 24.8.1998.
 Unesite drugi datum: 12.10.2010.
 Broj dana izmedju dva datuma je: 4440
```

Primer 4

```
Interakcija sa programom:
 Unesite prvi datum: 24.8.1998.
 Unesite drugi datum: 31.4.2004.
 Greska: neispravan unos.
```

Zadatak 1.7.29 Napisati funkciju `void romb(int n)` koja iscrtava romb čija je stranica dužine n. Napisati program koji učitava ceo pozitivan broj i iscrtava odgovarajuću sliku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj n: 5
     *****
    *****
   *****
  *****
 *****
```

Primer 2

```
Interakcija sa programom:
 Unesite broj n: 2
  **
 **
```

Primer 3

```
Interakcija sa programom:
 Unesite broj n: -5
 Greska: neispravan unos.
```

Zadatak 1.7.30 Napisati funkciju `void grafikon_h(int a, int b, int c, int d)` koja iscrtava horizontalni prikaz zadatih vrednosti. Napisati program koji učitava četiri pozitivna cela broja i iscrtava odgovarajuću sliku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve: 4 1 7 5
 ****
 *
 *******
 *****
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve: 5 2 2 10
 *****
 **
 **
 **********
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve: 8 -2 5 4
 Greska: neispravan unos.
```

Zadatak 1.7.31 Napisati funkciju `void grafikon_v(int a, int b, int c, int d)` koja iscrtava vertikalni prikaz zadatih vrednosti. Napisati program koji učitava četiri pozitivna cela broja i iscrtava odgovarajuću sliku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve: 4 1 7 5
   *
   *
   **
 * **
 * **
 * **
 ****
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve: 5 2 2 4
 *
 *  *
 *  *
 ****
 ****
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve: 8 -2 5 4
 Greska: neispravan unos.
```

1.8 Rešenja

Rešenje 1.7.1

```
#include <stdio.h>

/* Funkcija racuna minimum tri cela broja. Promenljive u listi
   argumenata funkcije (x, y i z), kao i one deklarisane u samoj
   funkciji (minimum), lokalne su za tu funkciju, sto znaci da im
   se ne moze pristupiti nigde izvan funkcije min. */
int min(int x, int y, int z) {
  /* Inicijalizacija minimuma na vrednost broja x. */
  int minimum = x;

  /* Poredjenje sa druga dva broja i po potrebi azuriranje
     vrednosti minimuma. */
  if (minimum > y)
    minimum = y;
  if (minimum > z)
    minimum = z;

  /* Vrednost minimuma se vraca kao povratna vrednost funkcije. */
  return minimum;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x, y, z;

  /* Ucitavanje vrednosti tri broja. */
  printf("Unesite brojeve: ");
  scanf("%d%d%d", &x, &y, &z);

  /* Poziv funkcije i ispis rezultata. */
  printf("Minimum: %d\n", min(x, y, z));

  return 0;
}
```

Rešenje 1.7.2

```
#include <stdio.h>
#include <math.h>

/* Funkcija vraca razlomljeni deo prosledjenog broja. */
float razlomljeni_deo(float x) {
  /* Napomena: Funkcija fabs racuna apsolutnu vrednost realnog
     broja i njena deklaracija se nalazi u zaglavlju math.h.
     Funkcija abs racuna apsolutnu vrednost celog broja i njena
     deklaracija se nalazi u zaglavlju stdlib.h. */
  x = fabs(x);
```

```
11
   /* Razlomljeni deo broja se dobija tako sto se od samog broja
13    oduzme njegov ceo deo. */
   return x - (int) x;
15 }

17 int main() {
   /* Deklaracija potrebne promenljive. */
19 float n;

21 /* Ucitavanje ulazne vrednosti. */
   printf("Unesite broj:");
23 scanf("%f", &n);

25 /* Ispis rezultata. */
   printf("Razlomljeni deo: %.6f\n", razlomljeni_deo(n));
27
   return 0;
29 }
```

Rešenje 1.7.3

```
1  #include <stdio.h>

3  /* Funkcija racuna zbir delilaca broja x. */
   int zbir_delilaca(int x) {
5    int i;

7    /* Inicijalizacija zbira na 0. */
     int zbir = 0;
9
     /* Svaki broj i izmedju 1 i sqrt(x) koji deli broj x se dodaje
11      zbiru. Ako je u pitanju broj za koji vazi da je i*i
        jednako x, onda se dodaje samo vrednost i. U suprotnom se
13      pored vrednosti i dodaje i x/i.
        Na primer, za x=6, kada je i=2, dodaju se i 2 i 6/2 = 3,
15      a za x = 4, kada je i=2, dodaje se samo 2. */
     for (i = 1; i * i <= x; i++) {
17     if (x % i == 0) {
         zbir += i;
19       if (i != x / i)
           zbir += x / i;
21     }
     }
23
     /* Povratna vrednost funkcije je dobijeni zbir. */
25   return zbir;
   }
27
   int main() {
29   /* Deklaracija potrebnih promenljivih. */
```

```
  int k, i;

  /* Ucitavanje broja k. */
  printf("Unesite broj k:");
  scanf("%d", &k);

  /* Provera ispravnosti ulaznih podataka. */
  if (k <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ispis zbira delilaca za svaki broj od 1 do k. */
  for (i = 1; i <= k; i++)
    printf("%d ", zbir_delilaca(i));
  printf("\n");

  return 0;
}
```

Rešenje 1.7.4

```
#include <stdio.h>

/* Funkcija za dva neoznacena broja x i n utvrdjuje da li je x neki
   stepen broja n. Ukoliko jeste, funkcija vraca izlozilac stepena,
   a u suprotnom vraca -1. */
int je_stepen(unsigned int x, unsigned int n) {
  /* Na pocetku, s = n^i = n^1 = n. */
  int i = 1;
  unsigned int s = n;

  /* U svakoj iteraciji petlje s se azurira tako da ima vrednost
     n^i. Postupak se ponavlja dok je s manji od x. */
  while (s < x) {
    s = s * n;
    i++;
  }

  /* Kako s ima vrednost n^i, ako vazi da je s jednako x, onda je
     bas brojac i trazeni izlozilac. */
  if (s == x)
    return i;

  /* Ako nije, onda se vraca -1. */
  return -1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int x, n;
```

```
  int stepen;

  /* Ucitavanje vrednosti x i n. */
  printf("Unesite dva broja: ");
  scanf("%u%u", &x, &n);

  /* Poziv funkcije. */
  stepen = je_stepen(x, n);

  /* U zavisnosti od povratne vrednosti funkcije, vrsi se ispis
     rezultata. */
  if (stepen != -1)
    printf("Jeste: %u=%u^%d\n", x, n, stepen);
  else
    printf("Broj %u nije stepen broja %u.\n", x, n);

  return 0;
}
```

Rešenje 1.7.5

```
#include <stdio.h>

/* Funkcija racuna nzd(x,y) primenom Euklidovog algoritma. */
int euklid(int x, int y) {
  int ostatak;
  /* Euklidov algoritam: trazi se nzd(x,y), npr. nzd(12,18).
     Postupak koji se primenjuje je sledeci:
     1. ostatak = x % y = 12 % 18 = 12.
     2. x postaje y => x = 18
     3. y postaje ostatak => y = 12 =>

     1. ostatak = x % y = 18 % 12 = 6
     2. x postaje y => x = 12
     3. y postaje ostatak => y = 6 =>

     1. ostatak = x % y = 12 % 6 = 0
     2. x postaje y => x = 6
     3. y postaje ostatak => y = 0

     Postupak se zavrsava kada y postane 0, a rezultat je
     poslednji ne-nula ostatak, tj. x. */
  while (y) {
    ostatak = x % y;
    x = y;
    y = ostatak;
  }

  /* Kao povratna vrednost funkcije se vraca x. */
  return x;
}
```

```
31
int main() {
33   /* Deklaracija potrebnih promenljivih. */
   int a, b;
35
   /* Ucitavanje vrednosti a i b. */
37   printf("Unesite dva cela broja:");
   scanf("%d%d", &a, &b);
39
   /* Ispis rezultata. */
41   printf("Najveci zajednicki delilac: %d\n", euklid(a, b));

43   return 0;
}
```

Rešenje 1.7.6 Pogledajte zadatak 1.7.3.

Rešenje 1.7.7

```
1 #include <stdio.h>

3 /* Funkcija broji koliko puta se realan broj x javlja u nizu unetih
     brojeva. */
5 int prebrojavanje(float x) {
   float y;
7   int broj_pojavljivanja = 0;

9   /* Brojevi se ucitavaju sve do unosa broja nula. Svaki put kada
       se unese broj koji je jednak broju x, brojac pojavljivanja se
11     uveca za 1. */
   printf("Unesite brojeve:\n");
13   while (1) {
     scanf("%f", &y);
15
     if (y == 0)
17     break;

19     if (x == y)
       broj_pojavljivanja++;
21   }

23   return broj_pojavljivanja;
}
25
int main() {
27   /* Deklaracije potrebnih promenljivih. */
   float x;
29   int rezultat;

31   /* Ucitavanje vrednosti broja x. */
   printf("Unesite broj x: ");
```

```
33   scanf("%f", &x);

35   /* Ucitavanje brojeva i racunanje rezultata. */
     rezultat = prebrojavanje(x);
37
     /* Ispis rezultata. */
39   printf("Broj pojavljivanja broja %.2f: %d\n", x, rezultat);

41   return 0;
}
```

Rešenje 1.7.8

```
1  #include <stdio.h>
   #include <math.h>
3
   /* Funkcija vraca 1 ako je broj prost, a 0 u suprotnom.  */
5  int prost(int x) {
     int i;
7
     /* Brojevi 2 i 3 su prosti. */
9    if (x == 2 || x == 3)
       return 1;
11
     /* Parni brojevi nisu prosti. */
13   if (x % 2 == 0)
       return 0;
15
     /* Ako se naidje na broj koji deli broj x, onda broj x nije
17      prost. Provera se vrsi za sve neparne brojeve izmedju 3 i
        sqrt(x), jer kada bi x imao parnog delioca, onda bi i broj 2
19      delio x, a taj uslov je vec proveren. */
     for (i = 3; i <= sqrt(x); i += 2)
21     if (x % i == 0)
         return 0;
23
     /* Ako nijedan od prethodnih uslova nije bio ispunjen, to znaci
25      da nijedan broj ne deli x, pa je on prost. */
     return 1;
27 }

29 /* Funkcija ispisuje prvih n prostih brojeva. Kljucna rec void
      oznacava da funkcija nema povratnu vrednost. */
31 void prvih_n_prostih(int n) {
     int broj_prostih = 0;
33   int k = 2;

35   /* Petlja se izvrsava sve dok se ne ispise n prostih brojeva. */
     while (broj_prostih < n) {
37     /* Ako se naidje na broj koji je prost, ispisuje se njegova
          vrednost i uvecava se brojac. */
```

```
39      if (prost(k)) {
          printf("%d ", k);
41        broj_prostih++;
        }
43
        /* Prelazi se na sledeci broj. */
45      k++;
        /* Napomena: Zbog prirode prostih brojeva, moze se krenuti i od
47         broja tri i vrsiti uvecavanje za dva, kako bi se preskocila
           provera parnih brojeva. */
49    }
      printf("\n");
51  }

53  /* Funkcija ispisuje sve proste brojeve cija je vrednost manja od
       n. */
55  void prosti_brojevi_manji_od_n(int n) {
      /* Ukoliko je n manje ili jednako 2, onda nema prostih brojeva
57       koji su manji od njega. U tom slucaju se ispisuje odgovarajuca
         poruka i naredbom return; se izlazi iz funkcije. */
59    if (n <= 2) {
        printf("ne postoje\n");
61      return;
      }
63
      /* Za svaki broj k izmedju 2 i n-1 se vrsi provera da li je prost
65       i ako jeste, ispisuje se njegova vrednost. */
      int k = 2;
67    while (k < n) {
        if (prost(k))
69        printf("%d ", k);
        k++;
71    }
      printf("\n");
73  }

75  int main() {
      /* Deklaracija potrebnih promenljivih. */
77    int n;

79    /* Ucitavanje broja n. */
      printf("Unesite broj n:");
81    scanf("%d", &n);

83    /* Provera ispravnosti ulaza. */
      if (n <= 0) {
85      printf("Greska: neispravan unos.\n");
        return 1;
87    }

89    /* Ispis rezultata. */
      printf("Prvih n prostih: ");
```

```
91    prvih_n_prostih(n);
      printf("Prosti manji od n: ");
93    prosti_brojevi_manji_od_n(n);

95    return 0;
}
```

Rešenje 1.7.10

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   /* Funkcija racuna aritmeticku sredinu cifara datog celog broja. */
5  float aritmeticka_sredina(int x) {
     /* Aritmeticka sredina broja 0 je 0. */
7    if (x == 0)
       return 0;
9
     /* Deklaracija i inicijalizacija brojaca. */
11   int zbir_cifara = 0;
     int broj_cifara = 0;
13
     /* Izracunava se apsolutna vrednost broja x kako bi program
15      ispravno radio i za negativne brojeve. */
     x = abs(x);
17
     /* Sve dok ima neobradjenih cifara, na zbir se dodaje poslednja
19      cifra, brojac cifara se uvecava za 1 i iz broja x se uklanja
        poslednja cifra. */
21   while (x) {
       zbir_cifara += x % 10;
23     broj_cifara++;
       x /= 10;
25   }

27   /* Kao povratna vrednost funkcije se vraca odgovarajuci
        kolicnik. */
29   return (float) zbir_cifara / broj_cifara;
}
31
int main() {
33   /* Deklaracija potrebne promenljive. */
     int x;
35
     /* Ucitavanje vrednosti broja x. */
37   printf("Unesite broj: ");
     scanf("%d", &x);
39
     /* Ispis rezultata. */
41   printf("Aritmeticka sredina: %.3f\n", aritmeticka_sredina(x));
```

```
43   return 0;
}
```

Rešenje 1.7.11

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija ispituje da li se cifra c nalazi u zapisu celog broja
   x. Vraca 1 ako je uslov ispunjen i 0 u suprotnom. */
int sadrzi(int x, int c) {
  /* Izracunava se apsolutna vrednost broja x. */
  x = abs(x);

  /* Izdvaja se cifra po cifra broja x. Ako se naidje na cifru cija
     je vrednost c, onda se kao rezultat funkcije vraca 1 (jer x
     sadrzi c). */
  while (x) {
    if (x % 10 == c)
      return 1;
    x /= 10;
  }

  /* Ako se petlja zavrsila, znaci da se nijednom nije naislo na
     cifru c, sto znaci da broj x ne sadrzi cifru c i kao povratna
     vrednost funkcije se vraca 0. */
  return 0;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int x, c;

  /* Ucitavanje vrednosti x i c. */
  printf("Unesite broj i cifru:");
  scanf("%d%d", &x, &c);

  /* Provera ispravnosti ulaza. */
  if (c < 0 || c > 9) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Racunanje i ispis rezultata. */
  if (sadrzi(x, c))
    printf("Cifra %d se nalazi u zapisu broja %d\n", c, x);
  else
    printf("Cifra %d se ne nalazi u zapisu broja %d\n", c, x);

  return 0;
}
```

Rešenje 1.7.12

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija odredjuje broj neparnih cifara u zapisu datog celog
   broja. */
int broj_neparnih_cifara(int x) {
  int brojac_neparnih = 0;
  char cifra;
  x = abs(x);

  while (x) {
    /* Izdvaja se poslednja cifra broja. */
    cifra = x % 10;

    /* Moze se izbeci koriscenje naredbe if pomocu narednog izraza.
       Naime, vrednost izraza cifra%2 je 1 kada je cifra neparna,
       odnosno 0 kada je parna. Tako ce na broj neparnih cifara
       biti dodata jednica ako je cifra neparna, a ako je parna
       bice dodata 0, sto jeste zeljeno ponasanje. */
    brojac_neparnih += (cifra % 2);
    x /= 10;
  }

  return brojac_neparnih;
}

int main() {
  /* Deklaracija potrebne promenljive. */
  int x;

  /* Ucitavanje brojeva sve do unosa broja nula i ispis
     broja neparnih cifara za svaki ucitani broj. */
  printf("Unesite cele brojeve:\n");
  while (1) {
    scanf("%d", &x);
    if (x == 0)
      break;

    printf("Broj neparnih cifara: %d\n", broj_neparnih_cifara(x));
  }

  return 0;
}
```

Rešenje 1.7.13

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija proverava da li su sve cifre broja x parne i vraca 1
   ako je uslov ispunjen i 0 ako nije. */
int sve_parne_cifre(int x) {
  char cifra;
  x = abs(x);

  /* Ako se naidje na cifru koja nije parna, onda se kao povratna
     vrednost funkcije vraca 0. */
  while (x > 0) {
    cifra = x % 10;
    if (cifra % 2 == 1)
      return 0;
    x /= 10;
  }

  /* Ako se doslo do kraja petlje, znaci da se nije naislo ni na
     jednu neparnu cifru, sto znaci da su sve cifre parne i da
     treba da se vrati 1. */
  return 1;
}

/* Funkcija proverava da li su sve cifre broja x jednake i vraca 1
   ako jesu, a 0 u suprotnom. */
int sve_cifre_jednake(int x) {
  char poslednja_cifra;
  x = abs(x);

  /* Izdvajanje poslednje cifre broja x. */
  poslednja_cifra = x % 10;
  x /= 10;

  /* Za sve ostale cifre se proverava da li su jednake poslednjoj.
     Ako se naidje na neku koja nije, onda nisu sve cifre broja x
     jednake i kao povratna vrednost se vraca 0. */
  while (x) {
    if (x % 10 != poslednja_cifra)
      return 0;

    x /= 10;
  }

  /* Ako se stiglo do kraja petlje, znaci da su sve cifre broja
     bile jednake poslednjoj cifri, pa se kao povratna vrednost
     vraca 1. */
  return 1;
}

```

```
int main() {
  /* Deklaracija potrebne promenljive. */
  int x;

  /* Ucitavanje broja x. */
  printf("Unesite broj:");
  scanf("%d", &x);

  /* U zavisnosti od povratne vrednosti napisanih funkcija vrsi se
     ispis odgovarajucih poruka. */
  if (sve_parne_cifre(x))
    printf("Sve cifre broja su parne.\n");
  else
    printf("Broj sadrzi bar jednu neparnu cifru.\n");

  if (sve_cifre_jednake(x))
    printf("Cifre broja su jednake.\n");
  else
    printf("Cifre broja nisu jednake.\n");

  return 0;
}
```

Rešenje 1.7.14

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/* Funkcija uklanja cifru sa pozicije p iz broja n. Cifra jedinica
   ima poziciju 1, desetica 2, itd. */
int ukloni(int n, int p) {
  int znak, tezina_pozicije, levi_deo, desni_deo;

  /* Racunanje znaka broja n. */
  znak = n < 0 ? 1 : -1;

  /* Racunanje apsolutne vrednosti broja n. */
  n = abs(n);

  /* Racunanje tezina prosledjene pozicije. */
  tezina_pozicije = pow(10, p - 1);

  /* Broj se deli na dva dela - deo levo od cifre koja se izbacuje
     i deo desno od cifre koja se izbacuje. */
  levi_deo = n / (10 * tezina_pozicije);
  desni_deo = n % tezina_pozicije;

  /* Povratna vrednost funkcije se dobija spajanjem levog i desnog
     dela i mnozenjem znakom pocetnog broja. */
  return znak * (levi_deo * tezina_pozicije + desni_deo);
```

```
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int broj, p;

  /* Ucitavanje vrednosti pozicije. */
  printf("Unesite poziciju: ");
  scanf("%d", &p);

  /* Provera ispravnosti ulaza. */
  if (p <= 0) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ucitavanje brojeva dok se ne unese nula i ispis brojeva
     dobijenih izbacivanjem cifre na poziciji p. */
  while (1) {
    printf("Unesite broj: ");
    scanf("%d", &broj);

    if (broj == 0)
      break;

    printf("Novi broj: %d\n", ukloni(broj, p));
  }

  return 0;
}
```

Rešenje 1.7.15

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija proverava da li se neka cifra nalazi u zapisu celog
   broja i ako se nalazi vraca odgovarajucu poziciju (tj. njenu
   tezinu koja je neki stepen broja 10), a u suprotnom vraca -1. Na
   primer, za broj = 1234 i cifra = 2, funkcija vraca 100. */
int pozicija_cifre(int broj, int cifra) {
  int tezina_pozicije = 1;

  while (broj) {
    if (broj % 10 == cifra)
      return tezina_pozicije;

    tezina_pozicije *= 10;
    broj /= 10;
  }
```

```
19    return -1;
}
21
/* Funkcija iz zapisa broja izbacuje cifru koja se nalazi na
23     prosledjenoj poziciji. Pozicija je stepen broja 10. Na primer,
     za x=1234 i pozicija = 10, treba da se izbaci 3.
25     levi_deo = 1234/(10*10) = 12
     desni_deo = 1234%10 = 4
27     Povratna vrednost je 12*10 + 4 = 124. */
int izbaci_cifru(int broj, int pozicija) {
29    int levi_deo = broj / (pozicija * 10);
    int desni_deo = broj % pozicija;
31    return levi_deo * pozicija + desni_deo;
}
33
/* Funkcija proverava da li su dva cela broja napisana pomocu istih
35     cifara. Vraca 1 ako je uslov ispunjen, a 0 u suprotnom. */
int zapis(int x, int y) {
37    int pozicija;
    x = abs(x);
39    y = abs(y);

41    while (x) {
      /* Provera da li y sadrzi poslednju cifru broja x. */
43      pozicija = pozicija_cifre(y, x % 10);

45      /* Ako ne sadrzi, x i y se ne zapisuju pomocu istih cifara. */
      if (pozicija == -1)
47        return 0;

49      /* Ako sadrzi, iz x se izbacuje poslednja cifra, a iz y se
           izbacuje ista ta cifra (koja se nalazi na pronadjenoj
51         poziciji. */
      x /= 10;
53      y = izbaci_cifru(y, pozicija);
    }
55
    /* Na kraju petlje iz x su izbacene sve cifre, a vazi da su
57       brojevi zapisani pomocu istih cifara samo ukoliko ni u y nema
       preostalih cifara. */
59    return y == 0;
}
61
int main() {
63    /* Deklaracija potrebnih promenljivih. */
    int x, y;
65
    /* Ucitavanje vrednosti x i y. */
67    printf("Unesite dva cela broja: ");
    scanf("%d%d", &x, &y);
69
    /* Ispis odgovarajuce poruke. */
```

```
71   if (zapis(x, y))
       printf("Uslov je ispunjen.\n");
73   else
       printf("Uslov nije ispunjen.\n");
75
     return 0;
77 }
```

Rešenje 1.7.16

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   /* Funkcija proverava da li se cifre u zapisu broja nalaze u
5     neopadajucem poretku. */
   int neopadajuce(int n) {
7    int tekuca_cifra, prethodna_cifra;
     n = abs(n);
9
     /* Izvan petlje se izdvaja poslednja cifra u zapisu broja da bi u
11      petlji mogla da se poredi sa sledecom. */
     prethodna_cifra = n % 10;
13   n /= 10;

15   /* U petlji se proverava poredak svake dve susedne cifre. Ukoliko
        se detektuje da je poredak narusen, izlazi se iz funkcije i
17      vraca se vrednost 0. */
     while (n) {
19     tekuca_cifra = n % 10;

21     if (tekuca_cifra > prethodna_cifra)
         return 0;
23
       /* Tekuca cifra postaje prethodna za narednu iteraciju. */
25     prethodna_cifra = tekuca_cifra;
       n /= 10;
27   }

29   /* Nakon izlaska iz petlje povratna vrednost funkcije je 1 jer u
        slucaju da je poredak u nekom trenutku narusen iz funkcije bi
31      se izaslo jos u petlji. */
     return 1;
33 }

35 int main() {
     /* Deklaracija potrebne promenljive. */
37   int n;

39   /* Ucitavanje vrednosti broja n. */
     printf("Unesite broj: ");
41   scanf("%d", &n);
```

```
  /* Ispis odgovarajuce poruke. */
  if (neopadajuce(n))
    printf("Cifre su u neopadajucem poretku.\n");
  else
    printf("Cifre nisu u neopadajucem poretku.\n");

  return 0;
}
```

Rešenje 1.7.17

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija proverava da li su cifre broja naizmenicno parne i
   neparne. Ako je uslov ispunjen vraca 1, u suprotnom vraca 0. */
int par_nepar(int x) {
  int prethodna_cifra, tekuca_cifra;
  x = abs(x);

  /* Poslednja cifra broja se izdvaja van petlje da bi u petlji
     moglo da se vrsi poredjenje. */
  prethodna_cifra = x % 10;
  x /= 10;

  while (x) {
    tekuca_cifra = x % 10;

    /* Ukoliko su uzastopne cifre iste parnosti, uslov nije
       ispunjen, rad petlje i funkcije se prekida i vraca se 0. */
    if (tekuca_cifra % 2 == prethodna_cifra % 2)
      return 0;

    /* Tekuca cifra postaje prethodna cifra za narednu iteraciju. */
    prethodna_cifra = tekuca_cifra;
    x /= 10;
  }

  /* Sve uzastopne cifre su razlicite parnosti jer ni jednom u
     petlji uslov da su cifre iste parnosti nije bio ispunjen. */
  return 1;
}

int main() {
  /* Deklaracija potrebne promenljive. */
  int n;

  /* Ucitavanje vrednosti broja n. */
  printf("Unesite broj n: ");
  scanf("%d", &n);
```

```
  /* Ispis odgovarajuce poruke. */
  if (par_nepar(n))
    printf("Broj ispunjava uslov.\n");
  else
    printf("Broj ne ispunjava uslov.\n");

  return 0;
}
```

Rešenje 1.7.18

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>

/* Funkcija racuna broj cifara celog broja n. */
int broj_cifara(int n) {
  int brojac = 0;
  n = abs(n);

  if (n < 10)
    return 1;

  while (n) {
    brojac++;
    n /= 10;
  }

  return brojac;
}

/* Funkcija racuna broj koji se dobija rotacijom broja n za jedno
   mesto ulevo. */
int rotacija(int n) {
  int znak, prva_cifra, n_bez_prve_cifre, br_cifara;

  znak = (n < 0) ? -1 : 1;
  n = abs(n);
  br_cifara = broj_cifara(n);

  /* Izdvajaju se prva cifra i deo broja bez prve cifre.
     Na primer: ako je n = 1234 onda je br_cifara = 4
     prva_cifra se dobija sa:
     n / (10 ^ (br_cifara - 1)) = 1234 / 1000 = 1.
     n_bez_prve_cifre se dobija sa: n % 1000 = 234. */
  int tezina_pozicije = pow(10, br_cifara - 1);
  prva_cifra = n / tezina_pozicije;
  n_bez_prve_cifre = n % tezina_pozicije;

  /* Rezultat se dobija nadovezivanjem prve cifre na kraj i
```

```
40      mnozenjem znakom pocetnog broja. */
    return znak * (n_bez_prve_cifre * 10 + prva_cifra);
42  }

44  int main() {
    /* Deklaracija potrebne promenljive. */
46    int n;

48    /* Ucitavanje brojeva sve do unosa broja nula i ispis brojeva
         dobijenih kao rezultat izvrsavanja funkcije rotacija nad
50       unetim brojevima. */
    while (1) {
52      printf("Unesite broj: ");
      scanf("%d", &n);
54
      if (n == 0)
56      break;

58      printf("Novi broj: %d\n", rotacija(n));
    }
60
    return 0;
62  }
```

Rešenje 1.7.19

```
1   #include <stdio.h>

3   /* Funkcija vraca zbir cifara datog broja x. */
    int zbir_cifara(int x) {
5     int zbir = 0;
      while (x) {
7       zbir += x % 10;
        x /= 10;
9     }
      return zbir;
11  }

13  /* Funkcija vraca 1 ako je broj srecan, a 0 u suprotnom. */
    int srecan(int x) {
15    /* Sve dok broj x ima vise od jedne cifre, vrednost broja x se
         zamenjuje zbirom njegovih cifara.
17       Na primer, pocetno x = 7698 nakon prve iteracije postaje
         x = 7+6+9+8 = 30, nakon druge iteracije postaje x = 3 + 0 = 3, a
19       zatim se izlazi iz petlje. */
      while (x >= 10)
21      x = zbir_cifara(x);

23    /* Broj je srecan ako na kraju x ima vrednost 1. */
      return (x == 1);
25  }
```

```
27 int main() {
     /* Deklaracija potrebnih promenljivih. */
29   int n, i;

31   /* Ucitavanje vrednosti broja n. */
     printf("Unesite broj n: ");
33   scanf("%d", &n);

35   /* Provera ispravnosti ulaza. */
     if (n <= 0) {
37     printf("Greska: neispravan unos.\n");
       return 1;
39   }

41   /* Ispis svih srecnih brojeva koji su manji ili jednaki n. */
     printf("Srecni brojevi: ");
43   for (i = 1; i <= n; i++)
       if (srecan(i))
45       printf("%d ", i);

47   printf("\n");
     return 0;
49 }
```

Rešenje 1.7.20

```
1  #include <stdio.h>
   #include <math.h>
3  #include <stdlib.h>

5  /* Funkcija racuna broj cifara celog broja n. */
   int broj_cifara(int n) {
7    int brojac = 0;
     n = abs(n);
9
     if (n < 10)
11     return 1;

13   while (n) {
       brojac++;
15     n /= 10;
     }
17
     return brojac;
19 }

21 /* Funkcija proverava da li je broj Armstrongov. */
   int armstrong(int x) {
23   int suma = 0;
     int n = broj_cifara(x);
```

```
  int x_pocetno = x;

  /* Racunanje suma n-tih stepena cifara broja x. */
  while (x) {
    suma += pow(x % 10, n);
    x /= 10;
  }

  /* Ako je suma jednaka pocetnoj vrednosti broja x, broj je
     Armstrongov, u suprotnom nije. */
  return x_pocetno == suma;
}

int main() {
  /* Deklaracija potrebne promenljive. */
  int x;

  /* Ucitavanje vrednosti broja x. */
  printf("Unesite broj: ");
  scanf("%d", &x);

  /* Ispis odgovarajuce poruke. */
  if (armstrong(x))
    printf("Broj je Armstrongov.\n");
  else
    printf("Broj nije Armstrongov.\n");

  return 0;
}
```

Rešenje 1.7.21

```
#include <stdio.h>
#include <math.h>

/* Funkcija racuna vrednost e^x kao parcijalnu sumu reda
   suma(x^n/n!), gde indeks n ide od od 0 do beskonacno, pri cemu
   se sumiranje sprovodi sve dok je sabirak po apsolutnoj vrednosti
   veci od date tacnosti eps. */
double e_na_x(double x, double eps) {
  double s = 1, clan = 1;
  int n = 1;

  /* Parcijalnu suma se formira tako sto se u svakoj iteraciji
     petlje promenljivoj s doda jedan sabirak sume oblika (x^n)/n!
     koji se cuva u promenljivoj clan.

     Svaki sabirak se dobija na osnovu prethodnog tako sto se
     prethodni pomnozi sa x i podeli sa n (n predstavlja redni broj
     sabirka u sumi).
```

```
20      Prvi sabirak (kome odgovara n=0) iznosi 1; zbog toga
        promenljive s i clan se inicijalizuju na vrednost 1.
22
        Sumiranje se sprovodi sve dok je sabirak po apsolutnoj
24      vrednosti veci od date tacnosti eps. */
    do {
26    clan = (clan * x) / n;
      s += clan;
28    n++;
    } while (fabs(clan) > eps);
30
    return s;
32  }

34  int main() {
    /* Deklaracija potrebnih promenljivih. */
36  double x, eps;

38  /* Ucitavanje vrednosti x i eps. */
    printf("Unesite broj x: ");
40  scanf("%lf", &x);
    printf("Unesite eps: ");
42  scanf("%lf", &eps);

44  /* Ispis rezultata. */
    printf("Rezultat: %f\n", e_na_x(x, eps));
46  return 0;
    }
```

Rešenje 1.7.22

```
    #include <stdio.h>
2   #include <math.h>

4   /* Funkcija ispisuje vrednosti funkcije sin(x) u n ravnomeno
       rasporedjenih tacaka na intervalu [a,b]. */
6   void ispis(float a, float b, int n) {
      double i;
8     double korak = (b - a) / (n - 1);

10    for (i = a; i <= b; i += korak, n--)
        printf("sin(%.4lf) = %.4lf \n", i, sin(i));
12
      /* Zapis realnih brojeva u racunaru ne mora da bude precizan
14       i sabiranje realnih brojeva moze, zbog akumuliranja greske,
         da dovede do toga da iako je
16       korak = (b - a) / (n - 1)
         ne vazi da je
18       b = a + korak + korak + ... + korak
         (gde se korak sabira n-1 puta). Vrednost ovog zbira, u
20       zavisnosti od konkretnih brojeva, moze da bude i malo veca
```

```
     i malo manja od b. Zbog toga, dodatno proveravamo da li
22     treba da stampamo vrednost funkcije i u tacki b */
  if(n != 0)
24    printf("sin(%.4lf) = %.4lf \n", b, sin(b));
}
26
int main() {
28  /* Deklaracije potrebnih promenljivih. */
  float a, b;
30  int n;

32  /* Ucitavanje granica intervala i provera ispravnosti ulaza. */
  printf("Unesite dva realna broja: ");
34  scanf("%f%f", &a, &b);
  if (b <= a) {
36    printf("Greska: neispravan unos.\n");
    return 1;
38  }

40  /* Ucitavanje broja n i provera ispravnosti ulaza. */
  printf("Unesite broj n: ");
42  scanf("%d", &n);
  if (n <= 1) {
44    printf("Greska: neispravan unos.\n");
    return 1;
46  }

48  /* Ispis rezultata. */
  printf("Rezultat:\n");
50  ispis(a, b, n);

52  return 0;
}
```

Rešenje 1.7.23

```
1 #include <stdio.h>

3 /* Funkcija vraca karakter koji se u engelskoj abecedi nalazi k mesta
      pre
    datog karaktera c. */
5 char sifra(char c, int k) {
  /* Provera da li je karakter malo slovo. */
7   if (c >= 'a' && c <= 'z') {
    /* Ako karakter koji je k pozicija pre datog karaktera ispada
9       iz opsega malih slova. */
    if (c - k < 'a')
11    /* Od k se oduzima rastojanje izmedju c i 'a' (jer je za
         toliko karaktera vec vraceno u nazad), kako bi se odredilo
13       koliko preostali broj karaktera koji treba preskociti od
         karaktera 'z'. */
```

```
15        return 'z' - (k - (c - 'a') - 1);
      else
17        /* U suprotnom, karakter c-k ne ispada iz opsega malih slova,
             te je dovoljno njega vratiti. */
19        return c - k;
    } else if (c >= 'A' && c <= 'Z') {
21      /* Postupak se ponavlja i za velika slova. */
      if (c - k < 'A')
23        return 'Z' - (k - (c - 'A') - 1);
      else
25        return c - k;
    }
27
    /* Ako nije ni malo ni veliko slovo, karakter se ne menja. */
29  return c;
}
31
int main() {
33  /* Deklaracije potrebnih promenljivih. */
    int k;
35  char c;

37  /* Ucitavanje vrednosti k. */
    printf("Unesite broj k: ");
39  scanf("%d", &k);

41  /* Ucitavanje karaktera sve do kraja ulaza i ispis njihove
       sifre. */
43  printf("Unesite tekst (CTRL + D za prekid): ");
    while ((c = getchar()) != EOF)
45    putchar(sifra(c, k));

47  return 0;
}
```

Rešenje 1.7.25

```
1 #include <stdio.h>

3 /* Funkcija proverava da li je godina prestupna. */
  int prestupna(int godina) {
5   if ((godina % 100 != 0 && godina % 4 == 0) || godina % 400 == 0)
      return 1;
7   else
      return 0;
9 }

11 /* Funkcija odredjuje broj dana u datom mesecu. */
   int broj_dana(int mesec, int godina) {
13   switch (mesec) {
     case 1:
```

```
15    case 3:
      case 5:
17    case 7:
      case 8:
19    case 10:
      case 12:
21      return 31;
      case 4:
23    case 6:
      case 9:
25    case 11:
        return 30;
27    case 2:
        if (prestupna(godina))
29        return 29;
        else
31        return 28;
    }
33  return -1;
}
35
/* Funkcija proverava da li je datum ispravan. Ako je datum
37     ispravan funkcija vraca 1, inace vraca 0. */
int ispravan(int dan, int mesec, int godina) {
39  /* Ako je godina negativna, datum nije ispravan. */
    if (godina < 0)
41    return 0;

43  /* Ako mesec nije u opsegu od 1 do 12, datum nije ispravan. */
    if (mesec < 1 || mesec > 12)
45    return 0;

47  /* Ako je dan manji od 1 ili veci od broja dana u datom mesecu,
       datum nije ispravan. */
49  if (dan < 1 || dan > broj_dana(mesec, godina))
      return 0;
51
    return 1;
53  }

55  /* Funkcija racuna sledeci dan. */
    void sledeci_dan(int dan, int mesec, int godina) {
57    /* Za kraj godine, odnosno za datum 31.12. sledeci datum je 1.1.
         i godina se uvecava za jedan. */
59    if (mesec == 12 && dan == 31)
        printf("1.1.%d.\n", godina + 1);
61    /* Ukoliko je dan jednak poslednjem danu u tom mesecu, odnosno
         ako je jednak broju dana u tom mesecu, onda je sledeci datum
63       kada se mesec uveca za 1, a dan postane 1. Bitan je redosled
         ovih naredbi. Ako bi ovo ispitivanje bilo prvo, onda bi se
65       mesec mogao uvecati na 13. sto ne bi bio ispravan datum. Zato
         se prvo proverava da li je kraj godine, pa tek onda da li je
```

```
67      kraj meseca. */
    else if (dan == broj_dana(mesec, godina))
69      printf("1.%d.%d.\n", mesec + 1, godina);
    /* Ako nije ni jedan od prethodna dva slucaja, onda se dan moze
71      uvecati na 1, bez bojazni da ce se prekoraciti broj dana u
        datom mesecu. */
73  else
      printf("%d.%d.%d.\n", dan + 1, mesec, godina);
75 }

77 int main() {
    /* Deklaracija potrebnih promenljivih. */
79  int dan, mesec, godina;

81  /* Ucitavanje vrednosti dana, meseca i godine. */
    printf("Unesite datum:");
83  scanf("%d.%d.%d.", &dan, &mesec, &godina);

85  /* Provera ispravnosti datuma. */
    if (!ispravan(dan, mesec, godina)) {
87    printf("Greska: neispravan unos.\n");
      return 1;
89  }

91  /* Poziv funkcije za ispis sledeceg dana. */
    printf("Datum sledeceg dana je:");
93  sledeci_dan(dan, mesec, godina);

95  return 0;
}
```

Rešenje 1.7.26

Za rešavanje ovog zadatka koristi se funkcija `od_nove_godine` koja je definisana u rešenju zadatka 1.7.28.

Rešenje 1.7.27

Za rešavanje ovog zadatka koristi se funkcija `do_kraja_godine` koja je definisana u rešenju zadatka 1.7.28.

Rešenje 1.7.28

```
#include <stdio.h>
2

/* Funkcija proverava da li je godina prestupna. */
4 int prestupna(int godina) {
    if ((godina % 100 != 0 && godina % 4 == 0) || godina % 400 == 0)
6     return 1;
    else
8     return 0;
}
```

```
10
   /* Funkcija odredjuje broj dana u datom mesecu. */
12 int broj_dana(int mesec, int godina) {
     switch (mesec) {
14   case 1:
     case 3:
16   case 5:
     case 7:
18   case 8:
     case 10:
20   case 12:
       return 31;
22   case 4:
     case 6:
24   case 9:
     case 11:
26     return 30;
     case 2:
28     if (prestupna(godina))
         return 29;
30     else
         return 28;
32   }
     return -1;
34 }

36 /* Funkcija proverava da li je datum ispravan. Ako je datum
      ispravan funkcija vraca 1, inace vraca 0. */
38 int ispravan(int dan, int mesec, int godina) {
     /* Ako je godina negativna, datum nije ispravan. */
40   if (godina < 0)
       return 0;
42
     /* Ako mesec nije u opsegu od 1 do 12, datum nije ispravan. */
44   if (mesec < 1 || mesec > 12)
       return 0;
46
     /* Ako je dan manji od 1 ili veci od broja dana u datom mesecu,
48      datum nije ispravan. */
     if (dan < 1 || dan > broj_dana(mesec, godina))
50     return 0;

52   return 1;
   }
54
   /* Funkcija odredjuje koliko dana je proteklo od pocetka godine. */
56 int od_nove_godine(int dan, int mesec, int godina) {
     int suma_dana = 0, i;
58
     /* Za sve mesece pre datog datuma dodaje se broj dana za dati
60      mesec. */
     for (i = 1; i < mesec; i++)
```

```
62      suma_dana += broj_dana(i, godina);

64    /* Na kraju se dodaje koliko je dana proteklo u datom mesecu, a
         to je zadato promenljivom dan. */
66    return suma_dana + dan;
  }
68
  /* Funkcija odredjuje koliko dana ima do kraja godine. */
70 int do_kraja_godine(int dan, int mesec, int godina) {
    int suma_dana = 0, i;
72
    /* Za sve mesece posle datog datuma dodaje se broj dana za dati
74       mesec. */
    for (i = mesec + 1; i <= 12; i++)
76    suma_dana += broj_dana(i, godina);

78  /* Na kraju se dodaje koliko je dana je ostalo u datom mesecu. */
    return suma_dana + broj_dana(mesec, godina) - dan;
80 }

82 /* Funkcija vraca 1 ako je prvi datum pre drugog datuma. U
     suprotnom vraca 0. */
84 int prethodi(int dan1, int mesec1, int godina1, int dan2,
                int mesec2, int godina2) {
86    if (godina1 < godina2)
      return 1;
88    else if (godina1 > godina2)
      return 0;
90    else if (mesec1 < mesec2)
      return 1;
92    else if (mesec1 > mesec2)
      return 0;
94    else if (dan1 < dan2)
      return 1;
96    else
      return 0;
98 }

100 /* Funkcija vraca broj dana u datoj godini. */
  int broj_dana_u_godini(int godina) {
102   if (prestupna(godina))
      return 366;
104   else
      return 365;
106 }

108 /* Funkcija racuna broj dana izmedju dva datuma. */
  int broj_dana_izmedju(int dan1, int mesec1, int godina1, int dan2,
110                       int mesec2, int godina2) {
    int pom, i;
112   int suma_dana = 0;
```

```
114   /* Provera koji od datuma je ranije. Ukoliko je potrebno,
        razmenjuju se vrednosti promenljivih tako da broj 1 ide uz raniji
         datum. */
116   if (!prethodi(dan1, mesec1, godina1, dan2, mesec2, godina2)) {
        pom = dan1;
118     dan1 = dan2;
        dan2 = pom;
120
        pom = mesec1;
122     mesec1 = mesec2;
        mesec2 = pom;
124
        pom = godina1;
126     godina1 = godina2;
        godina2 = pom;
128   }

130   /* Ako su godine razlicite. */
      if (godina1 != godina2) {
132     /* Za manji datum dodaje se broj dana do kraja godine. */
        suma_dana = do_kraja_godine(dan1, mesec1, godina1);
134
        /* Za sve godine koje su izmedju dve date godine dodaje se broj
136        dana u tim godinama. */
        for (i = godina1 + 1; i < godina2; i++)
138       suma_dana += broj_dana_u_godini(i);

140     /* Za veci datum dodaje se broj dana od pocetka godine. */
        suma_dana += od_nove_godine(dan2, mesec2, godina2);
142   }
      /* Ako su godine iste, ali meseci razliciti. */
144   else if (mesec1 != mesec2) {
        /* Dodaje se broj dana do kraja prvog meseca. */
146     suma_dana = broj_dana(mesec1, godina1) - dan1;

148     /* Dodaje se broj dana za svaki mesec koji je izmedju dva data
           meseca. Kako su godina1 i godina2 jednake svejedno je koja
150        od ove dve promenljive se koristi u pozivu funkcije. */
        for (i = mesec1 + 1; i < mesec2; i++)
152       suma_dana += broj_dana(i, godina1);

154     /* Dodaje se broj dana od pocetka meseca. */
        suma_dana += dan2;
156   }
      /* Ako su i godine i meseci jednaki. */
158   else
        suma_dana = dan2 - dan1;
160
      return suma_dana;
162 }

164 int main() {
```

```
  /* Deklaracija potrebnih promenljivih. */
  int dan1, mesec1, godina1, dan2, mesec2, godina2;

  /* Ucitavanje datuma. */
  printf("Unesite prvi datum:");
  scanf("%d.%d.%d.", &dan1, &mesec1, &godina1);

  printf("Unesite drugi datum:");
  scanf("%d.%d.%d.", &dan2, &mesec2, &godina2);

  /* Provera ispravnosti unetih datuma. */
  if (!ispravan(dan1, mesec1, godina1)
      || !ispravan(dan2, mesec2, godina2)) {
    printf("Greska: neispravan unos.\n");
    return 1;
  }

  /* Ispis rezultata. */
  printf("Broj dana izmedju dva datuma je: %d\n",
         broj_dana_izmedju(dan1, mesec1, godina1, dan2, mesec2,
                           godina2));

  return 0;
}
```

Rešenje 1.7.29

```
#include <stdio.h>

/* Funkcija iscrtava romb. */
void romb(int n) {
  int i, j;

  /* Petlja iscrtava liniju po liniju romba. */
  for (i = 0; i < n; i++) {
    /* Ispis n-i-1 praznina. */
    for (j = 0; j < n - i - 1; j++)
      printf(" ");

    /* Ispis n zvezdica. */
    for (j = 0; j < n; j++)
      printf("*");

    /* Prelazak u sledeci red. */
    printf("\n");
  }
}

int main() {
  /* Deklaracija potrebne promenljive. */
  int n;
```

```
25
  /* Ucitavanje vrednosti broja n. */
27  printf("Unesite broj n: ");
  scanf("%d", &n);
29
  /* Provera ispravnosti ulaza. */
31  if (n <= 0) {
    printf("Greska: neispravan unos.\n");
33    return 1;
  }
35
  /* Iscrtavanje romba. */
37  romb(n);

39  return 0;
}
```

Rešenje 1.7.30

```
1 #include <stdio.h>

3 /* Funkcija stampa n zvezdica za kojima sledi znak za novi red. */
  void stampaj_zvezdice(int n) {
5   int i;
    for (i = 0; i < n; i++)
7     printf("*");

9   printf("\n");
  }
11
  /* Funkcija crta grafikon. */
13 void grafikon_h(int a, int b, int c, int d) {
    /* Prvo se ispisuje a zvezdica. */
15  stampaj_zvezdice(a);

17  /* Postupak se ponavlja za vrednosti b, c i d. */
    stampaj_zvezdice(b);
19  stampaj_zvezdice(c);
    stampaj_zvezdice(d);
21 }

23 int main() {
    /* Deklaracija potrebnih promenljivih. */
25  int a, b, c, d;

27  /* Ucitavanje vrednosti a,b,c,d. */
    printf("Unesite brojeve: ");
29  scanf("%d%d%d%d", &a, &b, &c, &d);

31  /* Provera ispravnosti ulaza i ispis rezultata. */
    if (a < 0 || b < 0 || c < 0 || d < 0)
```

```
33      printf("Greska: neispravan unos.\n");
    else
35      grafikon_h(a, b, c, d);

37  return 0;
}
```

Rešenje 1.7.31

```
1  #include <stdio.h>

3  /* Funkcija racuna najveci od 4 prosledjena broja. */
   int maksimum(int a, int b, int c, int d) {
5    int maks;

7    maks = a;
     if (b > maks)
9      maks = b;
     if (c > maks)
11     maks = c;
     if (d > maks)
13     maks = d;

15   return maks;
   }
17
   /* Pomocna funkcija za ispis beline ili zvezdice. */
19 void ispisi_znak(int polje, int granica) {
     if (polje < granica)
21     printf(" ");
     else
23     printf("*");
   }
25
   /* Funkcija iscrtava vertikalni grafikon. */
27 void grafikon_v(int a, int b, int c, int d) {
     int i, maks;
29
     /* Pronalazak najvece od zadate cetiri vrednosti. */
31   maks = maksimum(a, b, c, d);

33   /* Grafikon ukupno ima maks horizontalnih linija. */
     for (i = 0; i < maks; i++) {
35     /* U svakoj od horizontalnih linija se nalazi po 4 polja: polje
          za a, b, c i d uspravnu liniju. U svako od polja treba da se
37        upise ili zvezdica ili praznina, u zavisnosti od vrednosti i
          toga koja linija se trenutno ispisuje. */
39
       /* Ispis znaka za polje a. */
41     ispisi_znak(i, maks - a);
```

```
43      /* Ispis znaka za polje b. */
        ispisi_znak(i, maks - b);
45
        /* Ispis znaka za polje c. */
47      ispisi_znak(i, maks - c);

49      /* Ispis znaka za polje d. */
        ispisi_znak(i, maks - d);
51
        /* Na kraju svake horizontalne linije stampa se novi red. */
53      printf("\n");
    }
55  }

57  int main() {
    /* Deklaracija potrebnih promenljivih. */
59  int a, b, c, d;

61  /* Ucitavanje vrednosti cetiri broja. */
    printf("Unesite brojeve: ");
63  scanf("%d%d%d%d", &a, &b, &c, &d);

65  /* Provera ispravnosti ulaza i poziv funkcije za ispis
       grafikona. */
67  if (a < 0 || b < 0 || c < 0 || d < 0) {
      printf("Greska: neispravan unos.\n");
69    return 1;
    } else
71    grafikon_v(a, b, c, d);

73  return 0;
    }
```

2

Napredni tipovi podataka

2.1 Nizovi

Zadatak 2.1.1 Napisati program koji učitava dimenziju niza, elemente niza i zatim ispisuje:

(a) elemente niza koji se nalaze na parnim pozicijama.

(b) parne elemente niza.

Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza:
 6
 Unesite elemente niza:
 1 8 2 -5 -13 75
 Elementi niza na parnim pozicijama:
 1 2 -13
 Parni elementi niza:
 8 2
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza:
 3
 Unesite elemente niza:
 11 81 -63
 Elementi niza na parnim pozicijama:
 11 -63
 Parni elementi niza:
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza:
 -4
 Greska: neispravan unos.
```

Zadatak 2.1.2 Napisati program koji učitava dimenziju niza, elemente niza i zatim menja uneti niz tako što kvadrira sve negativne elemente niza. Maksimalni

broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 6
 Unesite elemente niza:
 12.34 -6 1 8 32.4 -16
 Rezultujuci niz:
 12.34 36 1 8 32.4 256
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 9
 Unesite elemente niza:
 -8.25 6 17 2 -1.5 1 -7 2.65 -125.2
 Rezultujuci niz:
 68.0625 6 17 2 2.25 1 49 2.65 15675.04
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza:
 9.53 5 1 4.89
 Rezultujuci niz:
 9.53 5 1 4.89
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 104
 Greska: neispravan unos.
```

Zadatak 2.1.3 Ako su $a = (a_1, \ldots, a_n)$ i $b = (b_1, \ldots, b_n)$ vektori dimenzije n, njihov skalarni proizvod se definiše kao $a \cdot b = a_1 \cdot b_1 + \ldots + a_n \cdot b_n$. Napisati program koji računa skalarni proizvod dva vektora. Vektori se zadaju kao celobrojni nizovi sa najviše 100 elemenata. Program učitava dimenziju i elemente nizova, a na izlaz ispisuje vrednost skalarnog proizvoda. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju vektora: 5
 Unesite koordinate vektora a:
 8 -2 0 2 4
 Unesite koordinate vektora b:
 35 12 5 -6 -1
 Skalarni proizvod: 240
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju vektora: 3
 Unesite koordinate vektora a:
 -1 0 1
 Unesite koordinate vektora b:
 5 5 5
 Skalarni proizvod: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju vektora: 0
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju vektora: 1
 Unesite koordinate vektora a:
 -1
 Unesite koordinate vektora b:
 1
 Skalarni proizvod: -1
```

Zadatak 2.1.4 Napisati program koji učitava dimenziju niza, elemente niza, a potom i ceo broj k i ispisuje indekse elemenata koji su deljivi sa k. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 10 14 86 20
 Unesite broj k: 5
 Rezultat: 0 3
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 6 14 8 9
 Unesite broj k: 5
 U nizu nema elemenata koji su
 deljivi brojem 5.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 6
 Unesite elemente niza: 8 9 11 -4 8 11
 Unesite broj k: 2
 Rezultat: 0 3 4
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 6
 Unesite elemente niza: 1 2 3 4 5 6
 Unesite broj k: 0
 Greska: neispravan unos.
```

Zadatak 2.1.5 Autobusi su označeni rednim brojevima (počevši od 1) i u nizu se čuva vreme putovanja svakog autobusa u minutima. Međutim, zbog radova na putu između Požege i Užica, svi autobusi koji saobraćaju na tom potezu (autobusi označeni rednim brojevima od k do t) saobraćaju m minuta duže. Napisati program koji učitava broj autobusa n, n celih brojeva koji označavaju vreme putovanja tih autobusa i vrednosti k, t i m i ispisuje vreme putovanja svih autobusa nakon unetih izmena. Maksimalni broj autobusa je 200. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj autobusa: 8
 Unesite vreme putovanja:
 24 78 13 124 56 90 205 45
 Unesite vrednosti k, t i m:
 3 6 23
 Vreme putovanja nakon izmena:
 24 78 36 147 79 113 205 45
```

Primer 2

```
Interakcija sa programom:
 Unesite broj autobusa: 8
 Unesite vreme putovanja:
 24 78 13 124 56 90 205 45
 Unesite vrednosti k, t i m:
 3 15 3
 Greska: neispravan unos.
```

Zadatak 2.1.6 Napisati program koji za učitani ceo broj ispisuje broj pojavljivanja svake od cifara u zapisu tog broja. Uputstvo: *Za evidenciju broja pojavljivanja svake cifre pojedinačno, koristiti niz.*

Primer 1

```
Interakcija sa programom:
 Unesite ceo broj: 2355623
 U zapisu broja 2355623, cifra 2 se pojavljuje 2 puta
 U zapisu broja 2355623, cifra 3 se pojavljuje 2 puta
 U zapisu broja 2355623, cifra 5 se pojavljuje 2 puta
 U zapisu broja 2355623, cifra 6 se pojavljuje 1 puta
```

Primer 2

```
Interakcija sa programom:
 Unesite ceo broj: -39902
 U zapisu broja -39902, cifra 0 se pojaviljuje 1 puta
 U zapisu broja -39902, cifra 2 se pojaviljuje 1 puta
 U zapisu broja -39902, cifra 3 se pojaviljuje 1 puta
 U zapisu broja -39902, cifra 9 se pojaviljuje 2 puta
```

Zadatak 2.1.7 Napisati program koji učitava karaktere sve do unosa karaktera *, a zatim ih ispisuje u redosledu suprotnom od redosleda čitanja. Maksimalni broj karaktera je 500.

Primer 1

```
Interakcija sa programom:
 Unesite karakter: a
 Unesite karakter: 8
 Unesite karakter: 5
 Unesite karakter: Y
 Unesite karakter: I
 Unesite karakter: o
 Unesite karakter: ?
 Unesite karakter: *
 ? o I Y 5 8 a
```

Primer 2

```
Interakcija sa programom:
 Unesite karakter: g
 Unesite karakter: g
 Unesite karakter: 2
 Unesite karakter: 2
 Unesite karakter: )
 Unesite karakter: )
 Unesite karakter: *
 ) ) 2 2 g g
```

Zadatak 2.1.8 Napisati program koji učitava karaktere sve do kraja ulaza, a potom i izračunava koliko se puta u unetom tekstu pojavila svaka od cifara, svako malo slovo i svako veliko slovo. Ispisati broj pojavljivanja samo za karaktere koji su se u unetom tekstu pojavili barem jednom. Uputstvo: *Za evidenciju broja pojavljivanja cifara, malih i velih slova korisiti pojedinačne nizove.*

Primer 1

```
Interakcija sa programom:
 Unesite tekst:
 Mis je dobio grip.
 Karakter b se pojavljuje 1 puta
 Karakter d se pojavljuje 1 puta
 Karakter e se pojavljuje 1 puta
 Karakter g se pojavljuje 1 puta
 Karakter i se pojavljuje 3 puta
 Karakter j se pojavljuje 1 puta
 Karakter o se pojavljuje 2 puta
 Karakter p se pojavljuje 1 puta
 Karakter r se pojavljuje 1 puta
 Karakter s se pojavljuje 1 puta
 Karakter M se pojavljuje 1 puta
```

Primer 2

```
Interakcija sa programom:
 Unesite tekst:
 Programiranje 1 je zanimljivo!!
 Karakter 1 se pojavljuje 1 puta
 Karakter a se pojavljuje 3 puta
 Karakter e se pojavljuje 2 puta
 Karakter g se pojavljuje 1 puta
 Karakter i se pojavljuje 3 puta
 Karakter j se pojavljuje 3 puta
 Karakter l se pojavljuje 1 puta
 Karakter m se pojavljuje 2 puta
 Karakter n se pojavljuje 2 puta
 Karakter o se pojavljuje 2 puta
 Karakter r se pojavljuje 3 puta
 Karakter v se pojavljuje 1 puta
 Karakter z se pojavljuje 1 puta
 Karakter P se pojavljuje 1 puta
```

Zadatak 2.1.9 Napisati program koji učitava jednu liniju teksta i ispisuje koliko puta se pojavilo svako od slova engleske abecede u unetom tekstu. Ne praviti razliku između malih i velikih slova.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Tasi, tasi, TaNaNa i SVILENA marama.....
  a:9 b:0 c:0 d:0 e:1 f:0 g:0 h:0 i:4 j:0 k:0 l:1 m:2
  n:3 o:0 p:0 q:0 r:1 s:3 t:3 u:0 v:1 w:0 x:0 y:0 z:0
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Mihailo Petrovic Alas (6 maj 1868 - 8 jun 1943)
  a:4 b:0 c:1 d:0 e:1 f:0 g:0 h:1 i:3 j:2 k:0 l:2 m:2
  n:1 o:2 p:1 q:0 r:1 s:1 t:1 u:1 v:1 w:0 x:0 y:0 z:0
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Alan Matison Tjuring (London, 23. jun 1912 - Cesir, 7. jun 1954)
  a:3 b:0 c:1 d:1 e:1 f:0 g:1 h:0 i:3 j:3 k:0 l:2 m:1
  n:7 o:3 p:0 q:0 r:2 s:2 t:2 u:3 v:0 w:0 x:0 y:0 z:0
```

Zadatak 2.1.10 Takmičari na Beogradskom maratonu su označeni rednim brojevima počevši od 0. Vremena za koja su takmičari istrčali maraton izražena u minutima se zadaju nizom celih brojeva u kojem indeks elementa niza označava redni broj takmičara. Napisati sledeće funkcije za obradu navedenih podataka:

(a) `void ucitaj(int a[], int n)` koja učitava elemente niza a dimenzije n.

(b) `void ispisi(int a[], int n)` koja ispisuje elemente niza a dimenzije n.

(c) `int suma(int a[], int n)` koja računa i vraća ukupno vreme trčanja svih takmičara.

(d) `float prosek(int a[], int n)` koja računa i vraća prosečno vreme (aritmetičku sredinu) trčanja takmičara.

(e) `int maksimum(int a[], int n)` koja izračunava i vraća najduže vreme trčanja takmičara.

(f) `int pozicija_minimum(int a[], int n)` koja vraća redni broj pobednika Beogradskog maratona, tj. onog takmičara koji je najkraće trčao. U slučaju da ima više takvih takmičara, vratiti onog sa najmanjim rednim brojem.

Napisati program koji učitava podatke o rezultatima takmičara na maratonu i ispisuje učitane podatke, ukupno, prosečno i maksimalno vreme trčanja, kao i redni broj pobednika maratona. Maksimalni broj takmičara je 1000. U slučaju

neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza:
 5
 Unesite elemente niza: 140 126 170 220 130
 Vreme trcanja takmicara: 140 126 170 220 130
 Ukupno vreme: 786
 Prosecno vreme trcanja: 157.20
 Maksimalno vreme trcanja: 220
 Indeks pobednika: 1
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza:
 -5
 Greska: neispravan unos.
```

Zadatak 2.1.11 Napisati funkciju koja izračunava broj elemenata celobrojnog niza koji su manji od poslednjeg elementa niza. Napisati program koji učitava dimenziju niza i elemente niza, a zatim ispisuje broj elemenata koji zadovoljavaju pomenuti uslov. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 11 2 4 9
 Rezultat: 2
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 7
 Unesite elemente niza: 7 2 1 14 65 2 8
 Rezultat: 4
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 25 18 29 30 14
 Rezultat: 0
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: -45
 Greska: neispravan unos.
```

Zadatak 2.1.12 Napisati funkciju koja izračunava broj parnih elemenata celobrojnog niza koji prethode maksimalnom elementu niza. Napisati program koji učitava dimenziju niza i elemente niza, a zatim ispisuje broj elemenata koji prethode maksimalnom elementu. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 11 2 4 9
 Rezultat: 0
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 7
 Unesite elemente niza: 7 2 1 14 65 2 8
 Rezultat: 2
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 25 18 29 30 14
 Rezultat: 1
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 105
 Greska: neispravan unos.
```

Zadatak 2.1.13 Napisati funkciju `int zbir(int a[], int n, int i, int j)` koja računa zbir elemenata niza celih brojeva a dužine n od pozicije i do pozicije j. Napisati program koji učitava dimenziju niza, elemente niza i vrednosti i i j i zatim ispisuje zbir u datom opsegu. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 11 5 6 48 8
 Unesite vrednosti za i i j: 0 2
 Zbir je: 22
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 3
 Unesite elemente niza: -2 8 1
 Unesite vrednosti za i i j: 1 12
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 7
 Unesite elemente niza: -2 5 9 11 6 -3 -4
 Unesite vrednosti za i i j: 2 5
 Zbir je: 23
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 9 5 7 6
 Unesite vrednosti za i i j: 2 2
 Zbir je: 7
```

Zadatak 2.1.14 Napisati funkciju `float zbir_pozitivnih(float a[], int n, int k)` koja izračunava zbir prvih k pozitivnih elemenata realnog niza a dužine n. Napisati program koji učitava dimenziju niza, elemente niza i broj k, a zatim ispisuje zbir prvih k pozitivnih elemenata niza. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 2.34 1 -12.7 5.2 -8 -6.2 7 14.2
 Unesite vrednost k: 3
 Zbir je: 8.54
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 3
 Unesite elemente niza:
 -6.598 -8.14 -15
 Unesite vrednost k: 4
 Zbir je: 0.00
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 7
 Unesite elemente niza:
 -35.11 5.29 -1.98 12.1 12.2 -3.33 -4.17
 Unesite vrednost k: 15
 Zbir je: 29.59
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 3
 Unesite elemente niza:
 -0.11 5.29 -4.17
 Unesite vrednost k: -15
 Greska: neispravan unos.
```

Zadatak 2.1.15 Napisati funkciju koja menja niz tako što razmenjuje mesta najmanjem i najvećem elementu niza. Ukoliko se neki od ovih elemenata javlja više puta, uzeti u obzir prvo pojavljivanje. Napisati program koji učitava dimenziju niza, elemente niza, a zatim ispisuje izmenjeni niz. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 8 -2 11 19 4
 Rezultujuci niz:
 8 19 11 -2 4
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 10
 Unesite elemente niza:
 46 -2 51 8 -5 66 2 8 3 14
 Rezultujuci niz:
 46 -2 51 8 66 -5 2 8 3 14
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 145
 Greska: neispravan unos.
```

Zadatak 2.1.16 Napisati program koji vrši pretragu niza nadmorskih visina.

(a) Napisati funkciju koja proverava da li niz sadrži zadati broj m. Povratna vrednost funkcije je 1 ako je vrednost sadržana u nizu ili 0 ako nije.

(b) Napisati funkciju koja vraća vrednost prve pozicije na kojoj se nalazi element koji ima vrednost m ili -1 ukoliko element nije u nizu.

(c) Napisati funkciju koja vraća vrednost poslednje pozicije na kojoj se nalazi element koji ima vrednost m ili -1 ukoliko element nije u nizu.

Program učitava podatke o nadmorskim visinama i ceo broj m, a zatim ispisuje da li u nizu postoji podatak o unetoj nadmorskoj visini. Ukoliko postoji, ispisuje i poziciju prvog i poslednjeg pojavljivanja vrednosti m u nizu. Pozicije se broje od 0. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza:
 7
 Unesite podatke:
 800 1100 -200 1400 -200 1100 800
 Unesite vrednost m:
 1100
 Nadmorska visina 1100 se nalazi medju podacima.
 Pozicija prvog pojavljivanja: 1
 Pozicija poslednjeg pojavljivanja: 5
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza:
 -5
 Greska: neispravan unos.
```

Zadatak 2.1.17 Marko skuplja sličice za Svetsko prvenstvo u fudbalu. Marko je primetio da mu se neke sličice ponavljaju i rešio je da ih razmeni sa drugarima. Napisati funkciju `int duplikati(int a[], int n, int b[])` koja od niza a dimenzije n formira niz b koji sadrži sve različite elemente niza a koji se pojavljuju bar dva puta u nizu. Funkcija kao povratnu vrednost vraća dimenziju niza b. Napisati program koji učitava brojeve Markovih sličica i ispisuje sve duplikate. Maksimalni broj elemenata niza je 600. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza a:
 4 11 4 6 8 4 6 6
 Elementi niza b: 4 6
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 13
 Unesite elemente niza a:
 8 26 7 2 1 1 7 2 2 2 7 5 1
 Elementi niza b: 7 2 1
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 2
 Unesite elemente niza a:
 9 5
 Elementi niza b:
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 0
 Greska: neispravan unos.
```

Zadatak 2.1.18 Palindrom je tekst koji se isto čita i sa leve i sa desne strane. Napisati funkciju koja proverava da li je tekst zadat nizom karaktera palindrom (zanemariti razliku između malih i velikih slova). Napisati program koji učitava dužinu niza i niz karaktera, a zatim ispisuje da li je uneti tekst palindrom. Maksimalni broj elemenata niza je 200. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 15
 Unesite elemente niza:
 AnaVoliMilovana
 Niz jeste palindrom.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 26
 Unesite elemente niza:
 Zanimljivo je programirati!
 Niz nije palindrom.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 1
 Unesite elemente niza:
 a
 Niz jeste palindrom.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 226
 Greska: neispravan unos.
```

Zadatak 2.1.19 Napisati funkciju koja proverava da li su elementi celobrojnog niza uređeni neopadajuće. Napisati program koji učitava dimenziju niza, elemente niza, a zatim ispisuje da li je pomenuti uslov ispunjen. Maksimalni broj elemenata niza je 300. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 7
 Unesite elemente niza: -40 -8 -8 2 30 30 46
 Niz jeste uredjen neopadajuce.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 4 23 15 30
 Niz nije uredjen neopadajuce.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 1
 Unesite elemente niza: 5
 Niz jeste uredjen neopadajuce.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 304
 Greska: neispravan unos.
```

Zadatak 2.1.20 U celobrojnom nizu se čuvaju informacije o prodaji artikala jedne prodavnice. Svaki indeks niza označava jedan dan u mesecu, a elementi niza predstavljaju broj artikala koji se prodao tog dana. Napisati funkciju koja računa najdužu uzastopnu seriju dana za koju važi da broj prodatih artikala nije opao. Napisati program koji učitava broj dana u mesecu, broj prodatih artikala za svaki dan u mesecu i zatim ispisuje dužinu izračunate serije. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 30
 Unesite broj prodatih artikala:
 89 171 112 67 119 36 181 157
 49 96 73 116 21 172
 140 0 23 71 157 135 11 166 21
 56 56 87 103 183 148 174
 Duzina najduzeg neopadajuceg
 prodavanja je 6.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 31
 Unesite broj prodatih artikala:
 215 223 262 95 18 116 334 97
 146 146 19 314 270 115 21 40
 253 27 210 68 96 175 41 242
 98 163 8 218 107 102
 Duzina najduzeg neopadajuceg
 prodavanja je 3.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: -5
 Greska: neispravan unos.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 31
 Unesite broj prodatih artikala:
 -215 223 262 95 18 116 334 97
 146 146 19 314 -270 115 21 40
 253 27 210 68 96 175 41 242
 98 163 -8 218 107 102
 Greska: neispravan unos.
```

Zadatak 2.1.21 Napisati funkciju koja određuje dužinu najduže serije jednakih uzastopnih elemenata u datom nizu brojeva. Napisati program koji učitava dimenziju niza i elemente niza, a zatim ispisuje dužinu najduže serije jednakih elemenata niza. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 9 -1 2 2 2 2 80 -200
 Duzina najduze serije je 4.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 9 9 0 -3 -3 -3 -3 72
 Duzina najduze serije je 4.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza: 1 2 3 4 5 6 7 8
 Duzina najduze serije je 1.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 108
 Greska: neispravan unos.
```

Zadatak 2.1.22 Napisati funkciju koja određuje da li se jedan niz javlja kao (uzastopni) podniz drugog niza.

(a) Niz b je uzastopni podniz niza a ako su elementi niza b uzastopni elementi niza a.

(b) Niz b je podniz niza a ako je redosled pojavljivanja elemenata niza b u nizu a isti i ne nužno uzastopan.

Napisati program koji učitava dimenzije i elemente dvaju nizova, a zatim ispisuje

da li je drugi niz podniz prvog niza. Maksimalni broj elemenata nizova a i b je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 -4 2 7 90 -22 15 14 7
 Unesite dimenziju niza: 4
 Unesite elemente niza: 90 -22 15 14
 Elementi drugog niza cine
 uzastopni podniz prvog niza.
 Elementi drugog niza cine
 podniz prvog niza.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 -4 2 7 90 -22 15 14 7
 Unesite dimenziju niza: 4
 Unesite elemente niza: 2 7 15 7
 Elementi drugog niza ne cine
 uzastopni podniz prvog niza.
 Elementi drugog niza cine
 podniz prvog niza.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 -4 2 7 90 -22 15 14 7
 Unesite dimenziju niza: 4
 Unesite elemente niza: 90 -22 200 1
 Elementi drugog niza ne cine
 uzastopni podniz prvog niza.
 Elementi drugog niza ne
 cine podniz prvog niza.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 -4 2 7 90 -22 15 14 7
 Unesite dimenziju niza: 1
 Unesite elemente niza: 90
 Elementi drugog niza cine
 uzastopni podniz prvog niza.
 Elementi drugog niza cine
 podniz prvog niza.
```

Zadatak 2.1.23 Za celobrojni niz a dimenzije n kažemo da je *permutacija* ako sadrži sve brojeve od 1 do n.

(a) Napisati funkciju `void brojanje(int a[], int b[], int n)` koja na osnovu celobrojnog niza a dimenzije n formira niz b dimenzije n tako što i-ti element niza b odgovara broju pojavljivanja vrednosti i u nizu a.

(b) Napisati funkciju `int permutacija(int a[], int n)` koja proverava da li je zadati niz permutacija. Funkcija vraća vrednost 1 ako je svojstvo ispunjeno, odnosno 0 ako nije. Uputstvo: *Koristiti funkciju* ***brojanje***.

Napisati program koji učitava dimenziju niza i elemente niza i ispisuje da li je uneti niz permutacija. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 1 5 4 3 2
 Uneti niz je permutacija.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 6
 Unesite elemente niza: 2 3 3 1 1 5
 Uneti niz nije permutacija.
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju niza: 1
 Unesite elemente niza: 1
 Uneti niz je permutacija.
```

Primer 4

```
Interakcija sa programom:
 Unesite dimenziju niza: 101
 Greska: neispravan unos.
```

Zadatak 2.1.24 Napisati program koji učitava dva cela broja i proverava da li se uneti brojevi zapisuju pomoću istih cifara.

Primer 1

```
Interakcija sa programom:
 Unesite dva broja: 251 125
 Brojevi se zapisuju istim ciframa.
```

Primer 2

```
Interakcija sa programom:
 Unesite dva broja: 8898 9988
 Brojevi se ne zapisuju istim ciframa.
```

Primer 3

```
Interakcija sa programom:
 Unesite dva broja: -7391 1397
 Brojevi se zapisuju istim ciframa.
```

Primer 4

```
Interakcija sa programom:
 Unesite dva broja: -1 1
 Brojevi se zapisuju istim ciframa.
```

Zadatak 2.1.25 Napisati program koji vrši transformacije niza.

(a) Napisati funkciju koja obrće elemente niza.

(b) Napisati funkciju koja rotira niz ciklično za jedno mesto ulevo.

(c) Napisati funkciju koja rotira niz ciklično za k mesta ulevo.

Program učitava dimenziju niza, elemente niza i pozitivan ceo broj k, a zatim ispisuje niz koji se dobija nakon obrtanja početnog niza, niz koji se dobija rotiranjem tako dobijenog niza za jedno mesto ulevo i niz koji se dobija rotiranjem novodobijenog niza za k mesta ulevo. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju niza: 6
 Unesite elemente niza: 7 -3 11 783 26 -19
 Elementi niza nakon obrtanja:
 -17 28 785 13 -1 9
 Elementi niza nakon rotiranja za 1 mesto ulevo:
 28 785 13 -1 9 -17
 Unesite jedan pozitivan ceo broj: 3
 Elementi niza nakon rotiranja za 3 mesto ulevo:
 -1 9 -17 28 785 13
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza: 252
 Greska: neispravan unos.
```

Zadatak 2.1.26 Napisati funkciju `void ukrsti(int a[], int b[], int n, int c[])` koja formira niz c koji se dobija naizmeničnim raspoređivanjem elemenata nizova a i b, tj. $c = [a_0, b_0, a_1, b_1, \ldots, a_{n-1}, b_{n-1}]$. Napisati program koji učitava dimenziju i elemente dvaju nizova i ispisuje niz koji se dobija ukrštanjem unetih nizova. Maksimalni broj elemenata niza a i b je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju nizova: 5
 Unesite elemente niza a: 2 -5 11 4 8
 Unesite elemente niza b: 3 3 9 -1 17
 Rezultujuci niz:
 2 3 -5 3 11 9 4 -1 8 17
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju nizova: 105
 Greska: neispravan unos.
```

Zadatak 2.1.27 Napisati funkciju `void spoji(int a[], int b[], int n, int c[])` koja od nizova a i b dimenzije n formira niz c čija prva polovina odgovara elemetima niza b, a druga polovina elementima niza a, tj. $c = [b_0, b_1, \ldots, b_{n-1}, a_0, a_1, \ldots, a_{n-1}]$. Napisati program koji učitava dimenziju i elemente dvaju nizova i ispisuje niz koji se dobija spajanjem unetih nizova na pomenuti način. Maksimalni broj elemenata niza a i b je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju nizova: 3
 Unesite elemente niza a: 4 -8 32
 Unesite elemente niza b: 5 2 11
 Rezultujuci niz:
 5 2 11 4 -8 32
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju nizova: 145
 Greska: neispravan unos.
```

*** Zadatak 2.1.28** Napisati funkciju `void spoji_sortirano(int a[], int b[], int n, int c[])` koja od nizova a i b dimenzije n koji su uređeni neopadajuće formira niz c koji je uređen na isti način. Napisati program koji učitava dimenziju i elemente uređenih nizova a i b i ispisuje niz koji se dobija spajanjem ovih nizova na pomenuti način. Maksimalni broj elemenata niza a i b je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite dimenziju nizova: 5
 Unesite elemente sortiranog niza:
 2 11 28 40 63
 Unesite elemente sortiranog niza:
 -19 -5 5 11 52
 Rezultujuci niz:
 -19 -5 2 5 11 11 28 40 52 63
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju nizova: 3
 Unesite elemente sortiranog niza:
 -2 4 8
 Unesite elemente sortiranog niza:
 6 15 19
 Rezultujuci niz:
 -2 4 6 8 15 19
```

Primer 3

```
Interakcija sa programom:
 Unesite dimenziju nizova: 145
 Greska: neispravan unos.
```

Zadatak 2.1.29 Napisati funkciju `void promeni_redosled(int a[], int n)` koja menja redosled elementima niza a dimenzije n tako da se parni elementi niza nalaze na početku niza, a neparni na kraju. Napisati program koji učitava dimenziju niza i elemente niza i ispisuje niz koji je izmenjen na pomenuti način. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati

odgovarajuću poruku o grešci. NAPOMENA: *Ne koristiti pomoćne nizove.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 10
 Unesite elemente niza:
 -2 8 11 53 59 20 17 -8 3 14
 Rezultujuci niz:
 14 142 -6 -278 28 34 33 -69 -9 9
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 10
 Unesite elemente niza:
 9 142 -9 -278 -69 33 34 28 -6 14
 Rezultujuci niz:
 -2 8 14 -8 20 59 17 53 3 11
```

Zadatak 2.1.30 Napisati funkciju koja iz datog niza briše sve elemente koji su prosti brojevi. Funkcija kao povratnu vrednost treba da vrati broj elemenata niza nakon brisanja. Napisati program koji učitava dimenziju niza i elemente niza i ispisuje niz koji se dobija brisanjem pomenutih elemenata. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Zadatak rešiti uz korišćenje pomoćnog niza.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 11 5 6 48 8
 Rezultujuci niz: 6 48 8
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 11 5 19 21
 Rezultujuci niz: 21
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 5
 Unesite elemente niza: 12 18 9 31 7
 Rezultujuci niz: 12 18 9
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 5
 Unesite elemente niza: -2 15 -11 8 7
 Rezultujuci niz: 15 8
```

Zadatak 2.1.31 Napisati funkciju koja iz datog niza briše sve neparne elemente. Funkcija kao povratnu vrednost treba da vrati broj elemenata niza nakon brisanja. Napisati program koji učitava dimenziju niza i elemente niza i ispisuje niz koji se dobija brisanjem neparnih elemenata. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Zadatak rešiti bez korišćenja pomoćnog niza.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 4
 Unesite elemente niza:
 8 9 15 12
 Rezultujuci niz: 8 12
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 6
 Unesite elemente niza:
 21 5 3 22 19 188
 Rezultujuci niz: 22 188
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 4
 Unesite elemente niza: 133 129 121 101
 Rezultujuci niz:
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 8
 Unesite elemente niza:
 15 -22 -23 13 18 46 14 -31
 Rezultujuci niz: -22 18 46 14
```

Zadatak 2.1.32 Napisati funkciju koja iz datog niza briše sve elemente koji nisu deljivi svojom poslednjom cifrom. Izuzetak su elementi čija je poslednja cifra nula. Funkcija kao povratnu vrednost treba da vrati broj elemenata niza nakon brisanja. Napisati program koji učitava dimenziju niza i elemente niza i ispisuje niz koji se dobija brisanjem pomenutih elemenata. Maksimalni broj elemenata niza je 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Zadatak rešiti bez korišćenja pomoćnog niza.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 9
 Unesite elemente niza a:
 173 -25 23 7 17 25 34 61 -4612
 Rezultujuci niz: -25 7 25 61 -4612
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 0
 Greska: neispravan unos.
```

Zadatak 2.1.33 Napisati funkciju koja iz datog niza briše sve brojeve koji nisu deljivi svojim indeksom. Ne razmatrati da li je u novom nizu, nakon brisanja i pomeranja, element deljiv svojim indeksom. Funkcija kao povratnu vrednost treba da vrati broj elemenata niza nakon brisanja. Napisati program koji učitava dimenziju niza i elemente niza i ispisuje niz koji se dobija brisanjem pomenutih elemenata. Maksimalni broj elemenata niza je 700. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Nulti element niza treba zadržati jer nije dozvoljeno deljenje nulom. Zadatak rešiti bez korišćenja pomoćnog niza.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 10
 Unesite elemente niza:
 4 2 1 6 7 8 10 2 16 3
 Rezultujuci niz: 4 2 6 16
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 10
 Unesite elemente niza:
 -8 5 10 6 7 10 8 2 16 27
 Rezultujuci niz: -8 5 10 6 10 16 27
```

Zadatak 2.1.34 Korišćenjem nizova moguće je predstaviti skupove podataka. Napisati program koji demonstrira osnovne operacije nad skupovima (uniju, presek i razliku). Pomoću dva niza predstaviti dva skupa celih brojeva, a zatim ispisati njihovu uniju, presek i razliku. Maksimalni broj elemenata dva uneta niza je 500. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o

grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj elemenata niza a: 5
 Unesite elemente niza a: 1 2 3 4 5
 Unesite broj elemenata niza b: 3
 Unesite elemente niza b: 5 4 9
 Unija: 1 2 3 4 5 9
 Presek: 4 5
 Razlika: 1 2 3
```

Primer 2

```
Interakcija sa programom:
 Unesite broj elemenata niza a: 3
 Unesite elemente niza a: 11 4 -5
 Unesite broj elemenata niza b: 2
 Unesite elemente niza b: 18 9
 Unija: 11 4 -5 18 9
 Presek:
 Razlika: 11 4 -5
```

Primer 3

```
Interakcija sa programom:
 Unesite broj elemenata niza a: 6
 Unesite elemente niza a: 12 7 9 12 5 1
 Greska: skup ne moze imati duplikate.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj elemenata niza a: -2
 Greska: neispravan unos.
```

Zadatak 2.1.35 Da bi opsluživanje klijenata bilo efikasno i udobno, prilikom ulaska u banku svaki klijent dobija redni broj opslužiavanja. Redni brojevi se čuvaju u nizu, počinju od vrednosti 1 i iznova se generišu svakog radnog dana. Postoje i specijalni klijenti (npr. oni koji podižu stambeni kredit) koji mogu dobiti i negativni redni broj da bi se razlikovali od uobičajenih klijenata. Pomozite radniku obezbeđenja da lakše prati redosled opsluživanja klijenata.

(a) Napisati funkciju koja ubacuje redni broj klijenta x na kraj niza (klijenta koji je poslednji došao).

(b) Napisati funkciju koja ubacuje redni broj klijenta x na početak niza (klijenta koji će biti prvi uslužen, na primer, lica sa posebnim potrebama, trudnice ili stara lica).

(c) Napisati funkciju koja ubacuje redni broj klijenta x na poziciju k koju bira radnik obezbeđenja (manje prioritetna lica, recimo službena lica ili roditelji sa decom).

(d) Napisati funkciju koja izbacuje prvi redni broj iz niza (redni broj usluženog klijenta).

(e) Napisati funkciju koja izbacuje poslednji redni broj iz niza (redni broj klijenta koji je odustao jer je shvatio da ima mnogo klijenata ispred njega).

(f) Napisati funkciju koja izbacuje redni broj iz niza sa pozicije k (redni broj klijenta koji je odustao jer je dugo čekao).

Napisati program koji testira rad navedenih funkcija. Maksimalni broj klijenata u jednom danu je 2000. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite trenutni broj klijenata: 8
 Unesite niz sa rednim brojevima klijenata: 2 5 -2 16 33 19 8 11
 Unesite klijenta kojeg treba ubaciti u niz: 35
 Niz nakon ubacivanja klijenta: 2 5 -2 16 33 19 8 11 35
 Unesite prioritetnog klijenta kojeg treba ubaciti u niz: 36
 Niz nakon ubacivanja klijenta: 36 2 5 -2 16 33 19 8 11 35
 Unesite prioritetnog klijenta kojeg treba ubaciti u niz i njegovu poziciju: -6 2
 Niz nakon ubacivanja klijenta: 36 2 -6 5 -2 16 33 19 8 11 35
 Niz nakon odlaska klijenta: 2 -6 5 -2 16 33 19 8 11 35
 Niz nakon odlaska poslednjeg klijenta: 2 -6 5 -2 16 33 19 8 11
 Unesite redni broj klijenta koji je napustio red: -2
 Niz nakon odlaska klijenta: 2 -6 5 16 33 19 8 11
```

2.2 Rešenja

Rešenje 2.1.1

```
#include <stdio.h>
#include <stdlib.h>

/* Predprocesorska direktiva kojom se definise maksimalan broj
   elemenata niza. */
#define MAKS 100

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS];
  int n, i;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza:\n");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    /* Za izlazak iz programa moze da se koristi i funkcija exit.
       Argument EXIT_FAILURE oznacava da je doslo do neke greske
       pri izvrsavanju programa. Deklaracija ove funkcije se nalazi
       u zaglavlju stdlib.h. */
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  printf("Unesite elemente niza:\n");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);

  /* Ispis elemenata niza na parnim pozicijama. */
  printf("Elementi niza na parnim pozicijama:\n");
  for (i = 0; i < n; i += 2)
    printf("%d ", a[i]);
  printf("\n");

  /* Ispis parnih elemenata niza. */
  printf("Parni elementi niza:\n");
  for (i = 0; i < n; i++)
    if (a[i] % 2 == 0)
      printf("%d ", a[i]);
  printf("\n");

  /* Kada se funkciji exit prosledi EXIT_SUCCESS to znaci da se
     program uspesno zavrsio. Efekat je isti navodjenju return 0;
     naredbe na ovom mestu. */
  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.2

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

int main() {
  /* Deklaracija potrebnih promenljivih. */
  float brojevi[MAKS];
  int n, i;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  printf("Unesite elemente niza:\n");
  for (i = 0; i < n; i++)
    scanf("%f", &brojevi[i]);

  /* Ukoliko je i-ti element niza brojevi[i] negativan broj,
     kvadrira se tako sto se pomnozi samim sobom. */
  for (i = 0; i < n; i++)
    if (brojevi[i] < 0)
      brojevi[i] *= brojevi[i];

  /* Ispis novodobijenog niza. */
  printf("Rezultujuci niz: ");
  for (i = 0; i < n; i++)
    printf("%g ", brojevi[i]);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.3

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS], b[MAKS];
  int n, i, skalarni_proizvod;
```

```
11    /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
      printf("Unesite dimenziju vektora: ");
13    scanf("%d", &n);
      if (n <= 0 || n > MAKS) {
15      printf("Greska: neispravan unos.\n");
        exit(EXIT_FAILURE);
17    }

19    /* Ucitavanje koordinata vektora. */
      printf("Unesite koordinate vektora a: ");
21    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
23    printf("Unesite koordinate vektora b: ");
      for (i = 0; i < n; i++)
25      scanf("%d", &b[i]);

27    /* Racunanje skalarnog proizvoda po zadatoj formuli. */
      skalarni_proizvod = 0;
29    for (i = 0; i < n; i++)
        skalarni_proizvod += a[i] * b[i];
31
      /* Ispis rezultata. */
33    printf("Skalarni proizvod: %d\n", skalarni_proizvod);

35    exit(EXIT_SUCCESS);
  }
```

Rešenje 2.1.4

```
1   #include <stdio.h>
    #include <stdlib.h>
3
    #define MAKS 100
5
    int main() {
7     /* Deklaracija potrebnih promenljivih. */
      int brojevi[MAKS];
9     int n, i, k, indikator;

11    /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
      printf("Unesite dimenziju niza: ");
13    scanf("%d", &n);
      if (n <= 0 || n > MAKS) {
15      printf("Greska: neispravan unos.\n");
        exit(EXIT_FAILURE);
17    }

19    /* Ucitavanje elemenata niza. */
      printf("Unesite elemente niza: ");
21    for (i = 0; i < n; i++)
```

```
    scanf("%d", &brojevi[i]);

  /* Ucitavanje broja k i provera ispravnosti ulaza. */
  printf("Unesite broj k: ");
  scanf("%d", &k);
  if (k == 0) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Promenljiva koja cuva informaciju o tome da li u nizu
     postoji element koji je deljiv brojem k. */
  indikator = 0;

  /* Ukoliko je element niza deljiv brojem k, indikator se
     postavlja na 1 i ispisuje se indeks tog elementa. */
  for (i = 0; i < n; i++) {
    if (brojevi[i] % k == 0) {
      if(!indikator) {
        printf("Rezultat: ");
        indikator = 1;
      }
      printf("%d ", i);
    }
  }

  /* Ukoliko je indikator jednak nuli to znaci da ne postoji
     element u nizu koji je deljiv brojem k. */
  if (indikator == 0)
    printf("U nizu nema elemenata koji su deljivi brojem %d.\n", k);
  else
    printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.5

```
#include <stdio.h>
#include <stdlib.h>

/* Indeksiranje autobusa pocinje od 1, pa zato maksimalna
   dimenzija niza mora biti 201, a ne 200. */
#define MAKS 201

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, niz[MAKS], i;
  int k, t, m;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
```

```
  printf("Unesite broj autobusa: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje vremena putovanja. */
  printf("Unesite vreme putovanja:\n");
  for (i = 1; i <= n; i++)
    scanf("%d", &niz[i]);

  /* Ucitavanje rednih brojeva autobusa cije se vreme putovanja
     menja i vrednosti kasnjenja. */
  printf("Unesite vrednosti k, t i m:\n");
  scanf("%d%d%d", &k, &t, &m);

  /* Provera ispravnosti ulaza. */
  if (k <= 0 || k > n || t <= 0 || t > n || m < 0) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Azuriranje vremena putovanja. */
  for (i = k; i <= t; i++)
    niz[i] += m;

  /* Ispis rezultata. */
  printf("Vreme putovanja nakon izmena: ");
  for (i = 1; i <= n; i++)
    printf("%d ", niz[i]);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.6

```
#include <stdio.h>
#include <stdlib.h>

#define BROJ_CIFARA 10

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int x, x_original, cifra, i;

  /* Svaki element niza brojaci predstavlja brojac za jednu od
    cifara: brojac[0] predstavlja broj nula u zapisu broja x
    brojac[1] predstavlja broj jedinica u zapisu broja x ...
    brojac[9] predstavlja broj devetki u zapisu broja x.
```

```
14      Brojace je potrebno inicijalizovati na pocetku. */
    /* I nacin: */
16  int brojaci[BROJ_CIFARA];
    for(i=0; i<BROJ_CIFARA; i++)
18    brojaci[i] = 0;

20  /* II nacin: Inicijalizacija pri samoj deklaraciji.
      Na ovaj nacin su svi elementi niza brojaci inicijalizovani na
22    nule.
      int brojaci[BROJ_CIFARA] = {0}; */
24
    /* Ucitavanje celog broja. */
26  printf("Unesite ceo broj:\n");
    scanf("%d", &x);
28
    /* Cuvanje pocetne vrednosti zbog finalnog ispisa. */
30  x_original = x;
    x = abs(x);
32
    /* Obrada cifara. */
34  do {
      /* Izdvajanje krajnje desne cifre. */
36    cifra = x % 10;

38    /* Uvecavanje broja pojavljivanja izdvojene cifre. */
      brojaci[cifra]++;
40
      /* Prelazak na analizu sledece cifre. */
42    x /= 10;
    } while (x);
44
    /* Ispis informacija o ciframa koje se nalaze u zapisu broja x. */
46  for (i = 0; i < BROJ_CIFARA; i++)
      if (brojaci[i]) {
48      printf("U zapisu broja %d, cifra %d se pojaviljuje %d puta\n",
              x_original, i, brojaci[i]);
50    }

52  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.7

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  #define MAKS 100
5
  int main() {
7   /* Deklaracije potrebnih promenljivih. */
    char karakteri[MAKS];
```

```
  char c;
  int i, n;

  /* Ucitavanje karaktera sve do unosa zvezdice ili do prekoracenja
     maksimalnog broja karaktera. */
  for (i = 0; i < MAKS; i++) {
    printf("Unesite karakter: ");
    scanf("%c", &c);
    /* Citanje znaka za novi red nakon unetog karaktera. */
    getchar();

    /* Ukoliko je unet karakter '*' izlazi se iz petlje. */
    if (c == '*')
      break;

    /* Smestanje procitanog karaktera u niz. */
    karakteri[i] = c;
  }

  /* Broj unetih karaktera nakon izlaska iz petlje je i. */
  n = i;

  /* Ispis karaktera u obrnutom redosledu. */
  for (i = n - 1; i >= 0; i--)
    printf("%c ", karakteri[i]);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.8

```
#include <stdio.h>
#include <stdlib.h>

#define BROJ_CIFARA 10
#define DUZINA_ABECEDE 26

/* Pomocna funkcija za ispis elemenata niza. Vrednost n oznacava
   broj elemenata niza (moze imati vrednost 10 ili 26), a karakter
   c oznacava prvi karakter za datu kategoriju ('a' za mala slova,
   'A' za velika slova i '0' za cifre). */
void ispisi(int niz[], int n, char c) {
  int i;
  for (i = 0; i < n; i++)
    if (niz[i] != 0)
      printf("Karakter %c se pojavljuje %d puta\n", c + i, niz[i]);
}

/* Funkcija inicijalizuje niz postavljajuci vrednosti svih
   elemenata na nulu. */
```

```
void inicijalizuj(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    niz[i] = 0;
}

int main() {
  /* Deklaracije nizova brojaca za cifre, mala i velika slova. */
  int cifre[BROJ_CIFARA];
  int mala_slova[DUZINA_ABECEDE];
  int velika_slova[DUZINA_ABECEDE];

  /* Deklaracije pomocnih promenljivih. */
  int c;

  /* Inicijalizacije brojaca nulama. */
  inicijalizuj(cifre, BROJ_CIFARA);
  inicijalizuj(mala_slova, DUZINA_ABECEDE);
  inicijalizuj(velika_slova, DUZINA_ABECEDE);

  /* Ucitavanje karaktera sve do kraja ulaza. */
  printf("Unesite tekst:\n");
  while ((c = getchar()) != EOF) {
    if (c >= 'A' && c <= 'Z') {
      /* Ako je procitani karakter veliko slovo uvecava se broj
       pojavljivanja odgovarajuceg velikog slova. Indeks velikog
       slova u nizu se odredjuje oduzimanjem slova 'A'.
       Na taj nacin slovo 'A' ce imati indeks 0, slovo 'B' indeks
       1, itd.*/
      velika_slova[c - 'A']++;
    } else if (c >= 'a' && c <= 'z') {
    /* Ako je procitani karakter malo slovo uvecava se broj
       pojavljivanja odgovarajuceg malog slova. */
      mala_slova[c - 'a']++;
    } else if (c >= '0' && c <= '9') {
    /* Ako je procitani karakter cifra uvecava se broj
       pojavljivanja odgovarajuce cifre. */
      cifre[c - '0']++;
    }
  }

  /* Ispis trazenih informacija. */
  ispisi(cifre, BROJ_CIFARA, '0');
  ispisi(mala_slova, DUZINA_ABECEDE, 'a');
  ispisi(velika_slova, DUZINA_ABECEDE, 'A');

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.9

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define DUZINA_ALFABETA 26

/* Pomocna funkcija za ispis elemenata niza. */
void ispisi(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%c:%d  ", 'a' + i, niz[i]);
  putchar('\n');
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int c;
  int mala_slova[DUZINA_ALFABETA] = {0};

  /* Ucitavanje karaktera sve do kraja ulaza. */
  while ((c = getchar()) != EOF) {
    /* Ako je procitani karakter slovo, broj pojavljivanja slova se
       uvecava. Kako se zanemaruje velicina slova, svako slovo se
       pretvori u malo i potom se element na odgovarajucoj poziciji
       u nizu uveca. */
    if (isalpha(c))
      mala_slova[tolower(c) - 'a']++;
  }

  /* Ispis rezultata. */
  ispisi(mala_slova, DUZINA_ALFABETA);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.10

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 1000

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}
```

```
14 /* Funkcija ispisuje elemente niza. */
   void ispisi(int a[], int n) {
16   int i;
     for (i = 0; i < n; i++)
18     printf("%d ", a[i]);
     printf("\n");
20 }

22 /* Funkcija racuna sumu elemenata niza. */
   int suma(int a[], int n) {
24   int i, suma_elemenata = 0;

26   for (i = 0; i < n; i++)
       suma_elemenata += a[i];
28
     return suma_elemenata;
30 }

32 /* Funkcija racuna prosecnu vrednost elemenata niza. */
   float prosek(int a[], int n) {
34   int suma_elemenata = suma(a, n);
     return (float) suma_elemenata / n;
36 }

38 /* Funkcija izracunava maksimum elemenata niza. */
   int maksimum(int a[], int n) {
40   int i, najveci = a[0];

42   for (i = 1; i < n; i++)
       if (a[i] > najveci)
44       najveci = a[i];

46   return najveci;
   }
48
   /* Funkcija izracunava poziciju maksimalnog elementa u nizu. */
50 int pozicija_maksimuma(int a[], int n) {
     int i, pozicija_najveceg = 0;
52
     for (i = 1; i < n; i++)
54     if (a[i] > a[pozicija_najveceg])
         pozicija_najveceg = i;
56
     return pozicija_najveceg;
58 }

60 int main() {
     /* Deklaracija potrebnih promenljivih. */
62   int a[MAKS];
     int n;
64
```

```
  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza:");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(a, n);

  /* Ispis elemenata niza. */
  printf("Vreme trcanja takmicara: ");
  ispisi(a, n);

  /* Ispis ukupnog, prosecnog i maksimalnog vremena. */
  printf("Ukupno vreme: %d\n", suma(a, n));
  printf("Prosecno vreme trcanja: %.2f\n", prosek(a, n));
  printf("Maksimalno vreme trcanja: %d\n", maksimum(a, n));

  /* Ispis indeksa pobednika. */
  printf("Indeks pobednika: %d\n", pozicija_maksimuma(a, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.11 Pogledajte zadatak 2.1.10.

Rešenje 2.1.12

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* I nacin: Funkcija vraca poziciju najveceg elementa niza.
   Prolazi se kroz niz i ako se naidje na element cija je
   vrednost veca od trenutno najveceg elementa (a[pozicija]),
   vrsi se azuriranje pozicije trenutno najveceg.
   int pozicija_najveceg(int a[], int n) {
     int i, pozicija = 0;
       for(i=1; i<n; i++)
         if(a[i] > a[pozicija])
```

```
            pozicija = i;
23        return pozicija;
      }
25
    Funkcija vraca broj parnih elemenata niza koji prethode
27  maksimalnom elementu niza.
    int prebrojavanje(int a[], int n) {
29      int i;
        int pozicija_maksimuma = pozicija_najveceg(a,n);
31      int broj_parnih = 0;
        for (i = 0; i < pozicija_maksimuma; i++) {
33        if (a[i] % 2 == 0) {
           broj_parnih++;
35        }
        }
37      return broj_parnih;
    }
39 */

41 /* II nacin:
      Zadatak se moze resiti i jednim prolazom kroz niz. Ideja je da
43    se paralelno radi pretraga maksimalnog elementa i prebrojavanje
      parnih elemenata koji mu prethode.
45
      Ovo moze da se uradi sa dva brojaca parnih elemenata:
47    1. broj_parnih - brojac koji cuva broj parnih elemenata
      koji prethode trenutnom maksimumu
49    2. broj_parnih_izmedju - brojac koji cuva broj parnih elemenata
      koji se nalaze iza trenutnog maksimuma
51
      Svaki put kada se maksimum azurira, na broj parnih se doda broj
53    parnih koji se prebrojao izmedju dva azuriranja, a
      broj_parnih_izmedju se vraca na nulu. */
55 int prebrojavanje_jednim_prolazom(int a[], int n) {
    int i;
57  int pozicija_maksimuma = 0;
    int broj_parnih = 0;
59  int broj_parnih_izmedju = 0;

61  for (i = 0; i < n; i++) {
      if (a[i] > a[pozicija_maksimuma]) {
63      pozicija_maksimuma = i;
        broj_parnih += broj_parnih_izmedju;
65      broj_parnih_izmedju = 0;
      }
67
      if (a[i] % 2 == 0)
69      broj_parnih_izmedju++;
    }
71
    return broj_parnih;
73 }
```

```
int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS];
  int n;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(a, n);

  /* Ispis rezultata. */
  /* I nacin: printf("%d\n", prebrojavanje(a, n)); */

  /* II nacin: Jednim prolazom kroz niz. */
  printf("Rezultat: %d\n", prebrojavanje_jednim_prolazom(a, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.13

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* Funkcija sabira elemenate niza od pozicije i do pozicije j. */
int zbir(int a[], int i, int j) {
  int k, rezultat = 0;

  /* Obilazak elemenata niza koji pripadaju zadatom opsegu. */
  for (k = i; k <= j; k++)
    rezultat += a[k];

  return rezultat;
}
```

```
25 int main() {
     /* Deklaracije potrebnih promenljivih. */
27   int n, i, j;
     int a[MAKS];
29
     /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
31   printf("Unesite dimenziju niza: ");
     scanf("%d", &n);
33   if (n <= 0 || n > MAKS) {
       printf("Greska: neispravan unos.\n");
35     exit(EXIT_FAILURE);
     }
37
     /* Ucitavanje elemenata niza. */
39   ucitaj(a, n);

41   /* Ucitavanje vrednosti granica i provera ispravnosti ulaza. */
     printf("Unesite vrednosti za i i j: ");
43   scanf("%d%d", &i, &j);
     if (i < 0 || j < 0 || i > n - 1 || j > n - 1 || i > j) {
45     printf("Greska: neispravan unos.\n");
       exit(EXIT_FAILURE);
47   }

49   /* Ispis rezultata. */
     printf("Zbir je: %d", zbir(a, i, j));
51
     exit(EXIT_SUCCESS);
53 }
```

Rešenje 2.1.14

```
   #include <stdio.h>
2  #include <stdlib.h>

4  #define MAKS 100

6  /* Funkcija ucitava elemente niza dimenzije n. */
   void ucitaj(float a[], int n) {
8    int i;
     printf("Unesite elemente niza: ");
10   for (i = 0; i < n; i++)
       scanf("%f", &a[i]);
12 }

14 /* Funkcija racuna zbir prvih k pozitivnih elemenata niza. */
   float zbir_pozitivnih(float a[], int n, int k) {
16   int i;
     float zbir = 0;
18
```

```
  /* Obilazi se element po element niza. Postupak se zavrsava
     ukoliko se dodje do kraja niza ili ukoliko se sabere k
     pozitivnih elemenata. */
  for (i = 0; i < n && k > 0; i++)
    if (a[i] >= 0) {
      zbir += a[i];
      /* Umanjuje se brojac pozitivnih elemenata. */
      k--;
    }

  return zbir;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, k;
  float a[MAKS];

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(a, n);

  /* Ucitavanje broja k i provera ispravnosti ulaza. */
  printf("Unesite vrednost k: ");
  scanf("%d", &k);
  if (k < 0 || k > n) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ispis rezultata. */
  printf("Zbir je: %.2f\n", zbir_pozitivnih(a, n, k));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.15

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
```

```
7  void ucitaj(int a[], int n) {
     int i;
9    printf("Unesite elemente niza: ");
     for (i = 0; i < n; i++)
11     scanf("%d", &a[i]);
   }
13
   /* Funkcija ispisuje elemente niza dimenzije n. */
15 void ispisi(int a[], int n) {
     int i;
17   for (i = 0; i < n; i++)
       printf("%d ", a[i]);
19   printf("\n");
   }
21
   /* Funkcija razmenjuje najmanji i najveci element niza. */
23 void razmeni_min_max(int brojevi[], int n) {
     int i;
25   /* Najvecim, kao i najmanjim elementom niza, proglasava se nulti
        element niza. Pozicije najveceg i najmanjeg elementa se
27      postavljaju na 0. */
     int najveci = brojevi[0], najmanji = brojevi[0];
29   int pozicija_najveceg = 0, pozicija_najmanjeg = 0;

31   /* U prolazu kroz niz trazi se najveci i najmanji element i pamte
        se njihove pozicije. */
33   for (i = 1; i < n; i++) {
       if (brojevi[i] > najveci) {
35       najveci = brojevi[i];
         pozicija_najveceg = i;
37     }

39     if (brojevi[i] < najmanji) {
         najmanji = brojevi[i];
41       pozicija_najmanjeg = i;
       }
43   }

45   /* Zamenjuju se elementi na pozicijama pozicija_najmanjeg i
        pozicija_najveceg. */
47   brojevi[pozicija_najveceg] = najmanji;
     brojevi[pozicija_najmanjeg] = najveci;
49 }

51 int main() {
     /* Deklaracija potrebnih promenljivih. */
53   int brojevi[MAKS];
     int n;
55
     /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
57   printf("Unesite dimenziju niza: ");
     scanf("%d", &n);
```

```
59    if (n <= 0 || n > MAKS) {
        printf("Greska: neispravan unos.\n");
61      exit(EXIT_FAILURE);
      }
63
      /* Ucitavanje elemenata niza. */
65    ucitaj(brojevi, n);

67    /* Razmena najmanjeg i najveceg elementa. */
      razmeni_min_max(brojevi, n);
69
      /* Ispis rezultata. */
71    printf("Rezultujuci niz:\n");
      ispisi(brojevi, n);
73
      exit(EXIT_SUCCESS);
75  }
```

Rešenje 2.1.16

```
  #include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS 100

6 /* Funkcija ucitava elemente niza dimenzije n. */
  void ucitaj(int a[], int n) {
8   int i;
    printf("Unesite podatke: ");
10  for (i = 0; i < n; i++)
      scanf("%d", &a[i]);
12 }

14 /* Funkcija proverava da li niz sadrzi zadatu vrednost m. */
  int sadrzi(int a[], int n, int m) {
16  int i;
    /* Prolazi se kroz sve elemente niza i ukoliko se naidje na
18       element cija je vrednost jednaka m, kao povratna vrednost
         funkcije se vraca 1. */
20  for (i = 0; i < n; i++)
      if (a[i] == m)
22      return 1;

24  /* Ako se stigne do kraja niza, znaci da se broj m ne nalazi
       u nizu. */
26  return 0;
  }
28
  /* Funkcija vraca indeks prvog pojavljivanja elementa m u nizu a
30    ili -1 ukoliko se m ne nalazi u nizu a. */
  int prvo_pojavljivanje(int a[], int n, int m) {
```

```
32    int i;
      for (i = 0; i < n; i++)
34      if (a[i] == m)
          return i;
36
      /* Ako se stigne do kraja niza, znaci da se broj m ne nalazi
38       u nizu. */
      return -1;
40  }

42  /* Funkcija vraca indeks poslednjeg pojavljivanja elementa m u nizu
       a ili -1 ukoliko se m ne nalazi u nizu a. */
44  int poslednje_pojavljivanje(int a[], int n, int m) {
      int i;
46
      /* Polazi se od kraja niza i poredi se element po element sa
48       zadatim brojem m. */
      for (i = n - 1; i >= 0; i--)
50      if (a[i] == m)
          return i;
52
      /* Ako se stigne do pocetka niza, znaci da se broj m ne nalazi
54       u nizu. */
      return -1;
56  }

58  int main() {
      /* Deklaracije potrebnih promenljivih. */
60    int a[MAKS];
      int n, m, i;
62
      /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
64    printf("Unesite dimenziju niza: ");
      scanf("%d", &n);
66    if (n <= 0 || n > MAKS) {
        printf("Greska: neispravan unos.\n");
68      exit(EXIT_FAILURE);
      }
70
      /* Ucitavanje elemenata niza. */
72    ucitaj(a, n);

74    /* Ucitavanje vrednosti za pretragu. */
      printf("Unesite vrednost m:");
76    scanf("%d", &m);

78    /* Ispis rezultata pretrage. */
      if (sadrzi(a, n, m)) {
80      printf("Nadmorska visina %d se nalazi medju podacima.\n", m);

82      i = prvo_pojavljivanje(a, n, m);
        printf("Pozicija prvog pojavljivanja: %d\n", i);
```

```
84
     i = poslednje_pojavljivanje(a, n, m);
86   printf("Pozicija poslednjeg pojavljivanja: %d\n", i);
   } else
88   printf("Nadmorska visina %d se ne nalazi medju podacima.\n", m);

90 exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.17

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS 600
5
   /* Funkcija ucitava elemente niza dimenzije n. */
7  void ucitaj(int a[], int n) {
     int i;
9    printf("Unesite elemente niza a: ");
     for (i = 0; i < n; i++)
11     scanf("%d", &a[i]);
   }
13
   /* Funkcija ispisuje elemente niza dimenzije n. */
15 void ispisi(int niz[], int n) {
     int i;
17   for (i = 0; i < n; i++)
       printf("%d ", niz[i]);
19   printf("\n");
   }
21
   /* Funkcija proverava da li niz a dimenzije n sadrzi zadatu
23    vrednost x. Pretraga se vrsi od prosledjene pozicije. */
   int sadrzi(int niz[], int n, int od_pozicije, int x) {
25   int i;
     for (i = od_pozicije; i < n; i++)
27     if (niz[i] == x)
         return 1;
29
     return 0;
31 }

33 /* Funkcija formira niz b tako sto u njega ubacuje sve elemente
      niza a koji se u tom nizu pojavljuju bar dva puta. */
35 int duplikati(int a[], int n, int b[]) {
     /* Promenljiva j je brojac elemenata rezultujuceg niza. */
37   int i, j = 0;

39   /* Obilazi se element po element niza a. Trenutni element je
        duplikat ukoliko se javlja jos neki put u nizu a. Dovoljno je
```

```
41      gledati da li se nalazi iza tekuceg elementa jer ako se
        nalazi ispred, onda je on vec obradjen (i duplikat je
43      detektovan). Element a[i] se dodaje u niz duplikata ako vazi:
        1. a[i] je duplikat
45      2. a[i] se ne nalazi u nizu duplikata
        Provera sadrzi(a, n, i+1, a[i]) proverava prvi uslov.
47      Provera !sadrzi(b, j, 0, a[i]) proverava drugi uslov. */
    for (i = 0; i < n; i++)
49    if (sadrzi(a, n, i + 1, a[i]) && !sadrzi(b, j, 0, a[i])) {
        b[j] = a[i];
51      j++;
      }
53
    /* Povratna vrednost funkcije je duzina niza b. */
55  return j;
}
57
int main() {
59  /* Deklaracija potrebnih promenljivih. */
    int a[MAKS], b[MAKS];
61  int n_a, n_b;

63  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
    printf("Unesite broj n: ");
65  scanf("%d", &n_a);
    if (n_a <= 0 || n_a > MAKS) {
67    printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
69  }

71  /* Ucitavanje podataka o slicicama. */
    ucitaj(a, n_a);
73
    /* Popunjavanje niza b duplikatima niza a. */
75  n_b = duplikati(a, n_a, b);

77  /* Ispis rezultata. */
    printf("Elementi niza b: ");
79  ispisi(b, n_b);

81  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.18

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <ctype.h>

5 #define MAKS 200
```

```
/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(char niz[], int n) {
  int i;
  printf("Unesite elemete niza: ");
  for (i = 0; i < n; i++)
    scanf("%c", &niz[i]);
}

/* Funkcija proverava da li je niz karaktera palindrom. */
int je_palindrom(char niz[], int n) {
  int i, j;
  /* U petlji se porede elementi niz[0] i niz[n-1], zatim niz[1] i
    niz[n-2]
     itd. Ako se naidje na par elemenata koji se razlikuju, moze se
    zakljuciti da
     niz nije palindrom. */
  for (i = 0, j = n-1; i < j; i++, j--)
    if (tolower(niz[i]) != tolower(niz[j]))
      return 0;

  /* Izvrsila se cela petlja pa se moze zakljuciti da je niz
     palindrom. */
  return 1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char niz[MAKS];
  int n;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Preskace se novi red nakon unosa dimenzije. Ovo se radi jer
     sledi ucitavanje karaktera i bez ove linije, prvi karakter
     koji bi se upisao u niz bi bio novi red. */
  getchar();

  /* Ucitavanje elemenata niza. */
  ucitaj(niz, n);

  /* Ispis rezultata. */
  if (je_palindrom(niz, n))
    printf("Niz jeste palindrom.\n");
  else
    printf("Niz nije palindrom.\n");
```

```
57    exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.19

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 300

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* Funkcija proverava da li je niz uredjen neopadajuce. */
int uredjen_neopadajuce(int niz[], int n) {
  int i;
  for (i = 0; i < n - 1; i++)
    if (niz[i] > niz[i + 1])
      return 0;

  return 1;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int n, niz[MAKS];

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(niz, n);

  /* Ispis rezultata. */
  if (uredjen_neopadajuce(niz, n))
    printf("Niz jeste uredjen neopadajuce.\n");
  else
    printf("Niz nije uredjen neopadajuce.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.20

```
#include <stdio.h>
#include <stdlib.h>

/* Maksimalan broj dana u mesecu je 31, ali dani pocinju od 1, pa
   je potrebno odvojiti 32 mesta u nizu jer se nulti ne koristi. */
#define MAKS_DANA 32

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite broj prodatih artikala: ");
  for (i = 0; i < n; i++) {
    scanf("%d", &a[i]);
    if (a[i] < 0) {
      printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
    }
  }
}

/* Funkcija racuna duzinu najduzeg neopadajuceg podniza niza a. */
int najduzi_neopadajuci(int a[], int n) {
  int i;
  /* Na pocetku duzina trenutne serije i duzina maksimalne serije
     se inicijalizuju na 1. */
  int duzina_trenutne_serije = 1;
  int duzina_najduze_serije = 1;

  for (i = 1; i < n; i++) {
    /* Proverava se da li su uzastopni elementi u neopadajucem
       poretku. Ako je to slucaj uvecava se duzina serije, a
       ako nije, duzina trenutne serije se vraca na 1,
       kako bi se ispravno racunala duzina sledece serije. */
    if (a[i] >= a[i - 1])
      duzina_trenutne_serije++;
    else
      duzina_trenutne_serije = 1;

    /* Ukoliko je trenutna duzina serije veca od duzine do sada
       najduze serije, azurira se vrednost duzine najduze serije. */
    if (duzina_trenutne_serije > duzina_najduze_serije)
      duzina_najduze_serije = duzina_trenutne_serije;
  }

  return duzina_najduze_serije;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS_DANA], n;
```

```
52   /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
     printf("Unesite dimenziju niza: ");
54   scanf("%d", &n);
     if (n <= 0 || n > MAKS_DANA) {
56     printf("Greska: neispravan unos.\n");
       exit(EXIT_FAILURE);
58   }

60   /* Ucitavanje elemenata niza. */
     ucitaj(a, n);
62
     /* Ispis rezultata. */
64   printf("Duzina najduzeg neopadajuceg prodavanja je %d.\n",
            najduzi_neopadajuci(a, n));
66
     exit(EXIT_SUCCESS);
68 }
```

Rešenje 2.1.21

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS 100
5
   /* Funkcija ucitava elemente niza dimenzije n. */
7  void ucitaj(int a[], int n) {
     int i;
9    printf("Unesite elemente niza: ");
     for (i = 0; i < n; i++)
11     scanf("%d", &a[i]);
   }
13
   /* Funkcija vraca duzinu najduze serije jednakih elemenata niza. */
15 int najduza_serija(int a[], int n) {
     int i;
17   /* Na pocetku i duzina trenutne serije i duzina maksimalne serije
        se inicijalizuju na 1. */
19   int trenutna_serija = 1;
     int najduza_serija = 1;
21
     for (i = 1; i < n; i++) {
23     /* Proverava se da li su uzastopni elementi jednaki. Ako je to
          slucaj Uvecavanje duzina serije. Ako uzastopni elementi nisu
25        jednaki serija je prekinuta i vrednost duzine trenutne serije
          se postavlja ponovo na 1 da bi mogla da se racuna duzina
27        sledece serije. */
       if (a[i] == a[i - 1])
29       trenutna_serija++;
       else
```

```
31        trenutna_serija = 1;

33      /* Ukoliko je trenutna duzina serije veca od duzine do sada
           najduze serije, parametar za duzinu najduze serije se
35         postavlja na novu, vecu vrednost. */
      if (trenutna_serija > najduza_serija)
37        najduza_serija = trenutna_serija;
    }
39
    return najduza_serija;
41 }

43 int main() {
    /* Deklaracija potrebnih promenljivih. */
45  int n, a[MAKS];

47  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
    printf("Unesite dimenziju niza: ");
49  scanf("%d", &n);
    if (n <= 0 || n > MAKS) {
51    printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
53  }

55  /* Ucitavanje elemenata niza. */
    ucitaj(a, n);
57
    /* Ispis rezultata. */
59  printf("Duzina najduze serije je %d.\n", najduza_serija(a, n));

61  exit(EXIT_SUCCESS);
  }
```

Rešenje 2.1.22

```
  #include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS 100

6 /* Funkcija ucitava elemente niza dimenzije n. */
  void ucitaj(int niz[], int n) {
8   int i;
    printf("Unesite elemente niza: ");
10  for (i = 0; i < n; i++)
      scanf("%d", &niz[i]);
12 }

14 /* a) */
  int podniz_uzastopnih(int a[], int n, int b[], int m) {
16  int i, j;
```

```
18    /* Obilaze se elementi prvog niza. Svaki element prvog niza moze
         biti pocetak podniza, odnosno pocetak drugog niza. */
20    for (i = 0; i + m - 1 < n; i++) {
        /* Prolaze se elementi drugog niza. Za svaki element niza b
22         proverava se da li je jednak odgovarajucem elementu niza a.
           Za niz a razmatra se da li podniz pocinje od pozicije i.
24         Tako 0-ti element niza b je na poziciji i, 1. element je na
           poziciji i+1, 2. na poziciji i+2, ..., j-ti na poziciji i+j.
26         Ako uslov nije ispunjen, petlja se prekida i proverava se da
           li na sledecoj poziciji u nizu a pocinje podniz. */
28      for (j = 0; j < m; j++)
          if (a[i + j] != b[j])
30          break;
        /* Ako petlja nije prekinuta nakon ispitivanja, brojac za niz b
32         je jedanak dimenziji niza b, odnosno svi elementi niza b se
           uzastopno nalaze u nizu a. */
34      if (j == m)
          return 1;
36    }

38    /* Ukoliko niz b jeste uzastopni podniz uslov u petlji ce u nekom
         trenutku biti ispunjen i iz petlje i funkcije ce se izaci sa
40       return naredbom. Ipak, ako se to nije desilo i dalje se
         izvrsava funkcija, onda niz b nije uzastopni podniz. */
42    return 0;
  }
44
  /* b) */
46  int podniz(int a[], int n, int b[], int m) {
    int i, j;
48
    /* Obilaze se elementi niza a. */
50    for (i = 0, j = 0; i < n && j < m; i++) {
        /* Svaki put kada se naidje na element niza b, brojac za niz b
52         se uvecava i proverava se da li se sledeci element niza b
           nalazi u nizu a. */
54      if (a[i] == b[j])
          j++;
56    }

58    /* Ukoliko se pronadju svi elementi niza b u nizu a, onda je
         brojac za niz b jednak dimenziji niza b. U tom slucaju se
60       vraca vrednost 1, odnosno da niz jeste podniz. */
    return j == m;
62  }

64  int main() {
    /* Deklaracija potrebnih promenljivih. */
66    int n, a[MAKS];
    int m, b[MAKS];
68
```

```
  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata prvog niza. */
  ucitaj(a, n);

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &m);
  if (m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata drugog niza. */
  ucitaj(b, m);

  /* a) */
  if (podniz_uzastopnih(a, n, b, m))
    printf("Elementi drugog niza cine uzastopni podniz "
           "prvog niza.\n");
  else
    printf("Elementi drugog niza ne cine uzastopni podniz "
           "prvog niza.\n");

  /* b) */
  if (podniz(a, n, b, m))
    printf("Elementi drugog niza cine podniz prvog niza.\n");
  else
    printf("Elementi drugog niza ne cine podniz prvog niza.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.23

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int niz[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++) {
```

```
11      scanf("%d", &niz[i]);

13      /* Niz moze sadrzati elemente koji nisu u opsegu od 1 do n. U
           tom slucaju taj niz nije permutacija. */
15      if (niz[i] <= 0 || niz[i] > n) {
          printf("Uneti niz nije permutacija.\n");
17        exit(EXIT_SUCCESS);
        }
19    }
  }
21
  /* Funkcija prebrojava koliko puta se pojavljuje svaki element niza
23     a. */
  void brojanje(int a[], int b[], int n) {
25    int i;

27    /* Niz b se inicijalizuje tako sto se za svaki element postavi
         da se poljavuljuje 0 puta u nizu a. */
29    for (i = 1; i <= n; i++)
        b[i] = 0;
31
      /* Petljom se prolazi kroz niz a i za svaki element a[i] uvecava
33       se broj njegovog pojavljivanja u nizu b. Na primer, ako je
         a[3] = 7, onda treba uvecati broj pojavljivanja broja 7, a to
35       je b[7]++, sto se krace moze zapisati kao b[a[3]]++.
         Pretpostavlja se da je niz a dobro zadat, odnosno da su sve
37       njegove vrednosti u intervalu od 1 do n. */
      for (i = 0; i < n; i++)
39      b[a[i]]++;
  }
41
  /* Funkcija proverava da li je niz a permutacija. */
43 int permutacija(int a[], int n) {
      /* Niz b moze imati index MAKS (jer niz b se posmatra od 1 do
45       MAKS), pa zato njegova dimenzija mora biti za jedan veca. */
      int b[MAKS + 1];
47    int i;

49    /* Racunanje broja pojavljivanja svakog broja niza a. */
      brojanje(a, b, n);
51
      /* Ukoliko se svaki element niza a javlja tacno jednom u nizu a,
53       onda niz a jeste permutacija. Ovo svojstvo se proverava
         koriscenjem dobijenog niza b. */
55    for (i = 1; i <= n; i++)
        if (b[i] != 1)
57        return 0;

59    return 1;
  }
61
  int main() {
```

```
63    /* Deklaracija potrebnih promenljivih. */
      int a[MAKS], n;
65
      /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
67    printf("Unesite dimenziju niza: ");
      scanf("%d", &n);
69    if (n <= 0 || n > MAKS) {
        printf("Greska: neispravan unos.\n");
71      exit(EXIT_FAILURE);
      }
73
      /* Ucitavanje elemenata niza a. */
75    ucitaj(a, n);

77    /* Ispis rezultata. */
      if (permutacija(a, n))
79      printf("Uneti niz je permutacija.\n");
      else
81      printf("Uneti niz nije permutacija.\n");

83    exit(EXIT_SUCCESS);
    }
```

Rešenje 2.1.24

```
1   #include <stdio.h>
    #include <stdlib.h>
3
    #define BROJ_CIFARA 10
5
    /* Funkcija inicijalizuje niz postavljajuci vrednosti svih
7      elemenata na nulu. */
    void inicijalizuj(int niz[], int n) {
9     int i;
      for (i = 0; i < n; i++)
11      niz[i] = 0;
    }
13
    /* Funkcija izdvaja cifru po cifru broja i uvecava odgovarajuci
15     element niza koji odgovara brojacu za tu cifru. Na primer, za
       broj=1123, po zavrsetku ove funkcije niz[1] ce imati vrednost 2
17     jer se cifra 1 pojavljuje 2 puta, niz[2] i niz[3] ce imati
       vrednost 1, a svi ostali elementi niza ce imati vrednost 0. */
19  void analiza_cifara(int broj, int niz[]) {
      int c;
21
      /* Inicijalizacija svih brojaca na nule. */
23    inicijalizuj(niz, BROJ_CIFARA);

25    /* Uvecavanje odgovarajucih brojaca. */
      do {
```

```
27      c = broj % 10;
        niz[c]++;
29      broj /= 10;
    } while (broj);
31  }

33  int main() {
    /* Niz cifre_broja_x predstavlja brojace za cifre broja x.
35     Niz cifre_broja_y predstavlja brojace za cifre broja y. */
    int cifre_broja_x[BROJ_CIFARA], cifre_broja_y[BROJ_CIFARA];
37  int x, y, i, indikator;

39  /* Ucitavanje brojeva x i y. */
    printf("Unesite dva broja: ");
41  scanf("%d%d", &x, &y);

43  /* Za slucaj da su unete vrednosti negativne, posmatra se njihova
       apsolutna vrednost. Ovo je opravdano iz razloga sto se brojevi
45     x i -x zapisuju istim ciframa. */
    x = abs(x);
47  y = abs(y);

49  /* Popunjavaju se nizovi brojacima cifara. */
    analiza_cifara(x, cifre_broja_x);
51  analiza_cifara(y, cifre_broja_y);

53  /* Promenljiva indikator sluzi za pracenje da li su oba broja
       sastavljena od istih cifara. */
55  indikator = 1;

57  for (i = 0; i < BROJ_CIFARA; i++) {
      /* Ako se broj pojavljivanja cifre i u zapisu broja x razlikuje
59       od broja pojavljivanja cifre i u zapisu broja y, brojevi se
         ne zapisuju istim ciframa. Zato se vrednost indikatora moze
61       postaviti na 0 i prekinuti dalje uporedjivanje broja
         pojavljivanja. */
63    if (cifre_broja_y[i] != cifre_broja_x[i]) {
        indikator = 0;
65      break;
      }
67  }
    /* Ako je vrednost promenljive indikator ostala 1, to znaci da u
69     petlji nije pronadjena cifra koja se ne pojavljuje isti broj
       puta u zapisima brojeva x i y. Zato se moze zakljuciti da se
71     brojevi zapisuju istim ciframa. */
    if (indikator)
73    printf("Brojevi se zapisuju istim ciframa.\n");
    else
75    printf("Brojevi se ne zapisuju istim ciframa.\n");

77  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.25

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispisi(int a[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", a[i]);
  printf("\n");
}

/* Funkcija obrce elemente niza. */
void obrni(int a[], int n) {
  int t, i, j;

  /* Za niz a[0], a[1], ...., a[n-2], a[n-1] obrnuti niz je a[n-1],
     a[n-2], ...., a[1], a[0]. Zato je potrebno razmeniti vrednosti
     elemenata a[0] i a[n-1], a[1] i a[n-2], itd. i zaustaviti se
     kada je vrednost indeksa prvog elementa veca od vrednosti
     drugog elementa. */
  for (i = 0, j = n - 1; i < j; i++, j--) {
    t = a[i];
    a[i] = a[j];
    a[j] = t;
  }
}

/* Funkcija rotira niz ciklicno za jedno mesto u levo. */
void rotiraj_za_1(int a[], int n) {
  int i, prvi = a[0];

  /* Pomeranje preostalih elemenata niza za jedno mesto u levo. */
  for (i = 0; i < n - 1; i++)
    a[i] = a[i + 1];

  /* Poslednjem elementu se dodeljuje sacuvana vrednost prvog
     elementa. */
  a[n - 1] = prvi;
}
```

```
/* Funkcija rotira niz ciklicno za k mesta u levo. */
void rotiraj_za_k(int a[], int n, int k) {
  int i;

  /* Odredjuje se vrednost broja k koja je u opsegu od 0 do n-1
     kako bi se izbegla suvisna pomeranja. */
  k = k % n;

  /* Niz se rotira za jednu poziciju ulevo k puta. */
  for (i = 0; i < k; i++)
    rotiraj_za_1(a, n);
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS];
  int n, k;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(a, n);

  /* Obrtanje niza. */
  printf("Elementi niza nakon obrtanja:\n");
  obrni(a, n);
  ispisi(a, n);

  /* Rotiranje za jedno mesto u levo. */
  printf("Elementi niza nakon rotiranja za 1 mesto ulevo:\n");
  rotiraj_za_1(a, n);
  ispisi(a, n);

  /* Rotiranje za k mesta u levo. */
  printf("Unesite jedan pozitivan ceo broj:");
  scanf("%d", &k);
  if (k <= 0) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }
  rotiraj_za_k(a, n, k);
  printf("Elementi niza nakon rotiranja za %d mesto ulevo:\n", k);
  ispisi(a, n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.26

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    scanf("%d", &niz[i]);
}

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispisi(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", niz[i]);
  printf("\n");
}

/* Funkcija formira niz c ukrstanjem nizova a i b. */
void ukrsti(int a[], int b[], int n, int c[]) {
  int i, j;
  /* Formira se treci niz. Koriste se dva indeksa: indeks i
     pomocu kojeg se pristupa elementima nizova a i b i koji treba
     uvecati za 1 nakon svake iteracije i indeks j pomocu kojeg se
     pristupa elementima rezultujuceg niza c; s obzirom da se u
     svakoj iteraciji u niz c smestaju dva elementa, jedan iz niza
     a i jedan iz niza b, indeks j se uvecava za 2 nakon svake
     iteracije. */
  for (i = 0, j = 0; i < n; i++, j += 2) {
    c[j] = a[i];
    c[j + 1] = b[i];
  }
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS], b[MAKS], c[2 * MAKS];
  int n;

  /* Ucitavanje dimenzije nizova i provera ispravnosti ulaza. */
  printf("Unesite dimenziju nizova: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata nizova. */
```

```
51    printf("Unesite elemente niza a: ");
      ucitaj(a, n);
53    printf("Unesite elemente niza b: ");
      ucitaj(b, n);
55
      /* Formiranje niza c. */
57    ukrsti(a, b, n, c);

59    /* Ispis elemenata rezultujuceg niza. */
      printf("Rezultujuci niz:\n");
61    ispisi(c, 2 * n);

63    exit(EXIT_SUCCESS);
  }
```

Rešenje 2.1.27

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS 100
5
   /* Funkcija ucitava elemente niza dimenzije n. */
7  void ucitaj(int niz[], int n) {
     int i;
9    for (i = 0; i < n; i++)
       scanf("%d", &niz[i]);
11 }

13 /* Funkcija ispisuje elemente niza dimenzije n. */
   void ispisi(int niz[], int n) {
15   int i;
     for (i = 0; i < n; i++)
17     printf("%d ", niz[i]);
     printf("\n");
19 }

21 /* Funkcija formira niz c nadovezivanjem nizova a i b. */
   void spoji(int a[], int b[], int n, int c[]) {
23   int i;

25   /* Niz c ima 2*n elemenata: prvih n elemenata su elementi niza b,
        a narednih n elemenata elementi niza a. Elementi niza b se
27      nalaze na pozicijama 0,1,2,...n-1, a elementi niza a na
        pozicijama n,n+1,...2*n-1. Jednim prolaskom kroz petlju na
29      poziciju i u nizu c se postavlja element b[i] niza b, a na
        poziciju n+i element a[i] niza a. */
31   for (i = 0; i < n; i++) {
       c[i] = b[i];
33     c[n + i] = a[i];
     }
```

```
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a[MAKS], b[MAKS], c[2 * MAKS];
  int n;

  /* Ucitavanje dimenzije nizova i provera ispravnosti ulaza. */
  printf("Unesite dimenziju nizova: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata nizova. */
  printf("Unesite elemente niza a: ");
  ucitaj(a, n);
  printf("Unesite elemente niza b: ");
  ucitaj(b, n);

  /* Formiranje niza c. */
  spoji(a, b, n, c);

  /* Ispis elemenata rezultujuceg niza. */
  printf("Rezultujuci niz:\n");
  ispisi(c, 2 * n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.28

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int niz[], int n) {
  int i;
  printf("Unesite elemente sortiranog niza:\n");
  for (i = 0; i < n; i++)
    scanf("%d", &niz[i]);
}

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispisi(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", niz[i]);
```

```
19    printf("\n");
  }
21
  int main() {
23    /* Deklaracija potrebnih promenljivih. */
      int a[MAKS], b[MAKS], c[2 * MAKS];
25    int n;
      /* Brojac u petlji za elemente niza a. */
27    int i = 0;
      /* Brojac u petlji za elemente niza b. */
29    int j = 0;
      /* Brojac u petlji za elemente niza c. */
31    int k = 0;

33    /* Ucitavanje dimenzije nizova i provera ispravnosti ulaza. */
      printf("Unesite dimenziju nizova: ");
35    scanf("%d", &n);
      if (n <= 0 || n > MAKS) {
37      printf("Greska: neispravan unos.\n");
        exit(EXIT_FAILURE);
39    }

41    /* Ucitavanje elemenata nizova. */
      ucitaj(a, n);
43    ucitaj(b, n);

45    /* Spajanje nizova. */
      while (i < n && j < n) {
47      /* Porede se elementi nizova a i b i u niz c upisuje se samo
           onaj koji je manji. Ako je upisan element iz niza a, onda se
49         vrsi i uvecavanje brojaca i (prelazak na sledeci element niza
           a), a ako je upisan element iz niza b, onda se vrsi
51         uvecavanje brojaca j (prelazak na sledeci element niza b). */
        if (a[i] < b[j]) {
53        c[k] = a[i];
          i++;
55      } else {
          c[k] = b[j];
57        j++;
        }
59
        /* U nizu c na poziciju k je upisan ili a[i] ili b[j]. Brojac k
61         se uvecava. */
        k++;
63    }

65    /* Ukoliko je ostalo elemenata u nizu a, upisuju se u niz c. */
      while (i < n) {
67      c[k] = a[i];
        k++;
69      i++;
      }
```

```
71
  /* Ukoliko je ostalo elemenata u nizu b, upisuju se u niz c. */
73  while (j < n) {
    c[k] = b[j];
75    k++;
    j++;
77  }

79  /* Ispis elemenata niza c cija dimenzija je zbir dimenzija nizova
       a i b. */
81  printf("Rezultujuci niz:\n");
  ispisi(c, 2 * n);
83
  exit(EXIT_SUCCESS);
85 }
```

Rešenje 2.1.29

```
#include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS 100

6 /* Funkcija ucitava elemente niza dimenzije n. */
  void ucitaj(int a[], int n) {
8   int i;
    printf("Unesite elemente niza: ");
10  for (i = 0; i < n; i++)
      scanf("%d", &a[i]);
12 }

14 /* Funkcija ispisuje elemente niza dimenzije n. */
  void ispisi(int niz[], int n) {
16  int i;
    for (i = 0; i < n; i++)
18    printf("%d ", niz[i]);
    printf("\n");
20 }

22 /* Funkcija razmenjuje elemente niza tako da se na pocetku niza
      nalaze svi parni elementi niza, a zatim svi neparni elementi
24    niza. */
  void promeni_redosled(int niz[], int n) {
26  int i = 0, j = n - 1, pom;

28  /* Krece se od pocetka niza (po brojacu i) i od kraja niza (po
       brojacu j) i svaki put kada se naidje na elemente koji po
30     parnosti ne odgovaraju delu niza u kome treba da budu,
       zamene se njihove vrednosti. */
32  while (i < j && i < n && j >= 0) {
      if (niz[i] % 2 != 0 && niz[j] % 2 == 0) {
```

```
34        pom = niz[i];
          niz[i] = niz[j];
36        niz[j] = pom;
      }
38
      /* Ukoliko je element na poziciji i paran, Prelazak na
40       sledeci element niza, brojac i se uvecava. */
      if (niz[i] % 2 == 0)
42      i++;

44    /* Ukoliko je element na poziciji j neparan, Prelazak na
         sledeci element niza, brojac j se smanjuje. */
46    if (niz[j] % 2 != 0)
        j--;
48  }
}
50
int main() {
52  /* Deklaracija potrebnih promenljivih. */
  int niz[MAKS];
54  int n;

56  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
58  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
60    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
62  }

64  /* Ucitavanje elemenata niza. */
  ucitaj(niz, n);
66
  /* Izmena niza na trazeni nacin. */
68  promeni_redosled(niz, n);

70  /* Ispis rezultata. */
  printf("Rezultujuci niz:\n");
72  ispisi(niz, n);

74  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.30

```
#include <stdio.h>
2 #include <stdlib.h>
#include <math.h>
4
#define MAKS 100
6
```

```
/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispisi(int a[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", a[i]);
  printf("\n");
}

/* Funkcija vraca 1 ako je broj prost, a 0 u suprotnom.  */
int prost(int x) {
  int i;

  /* Brojevi 2 i 3 su prosti. */
  if (x == 2 || x == 3)
    return 1;

  /* Parni brojevi nisu prosti. */
  if (x % 2 == 0)
    return 0;

  /* Ako se naidje na broj koji deli broj x, onda broj x nije
     prost. Provera se vrsi za sve neparne brojeve izmedju 3 i
     korena broja x, jer kada bi x imao parnog delioca, onda bi
     i broj 2 delio x, a taj uslov je vec proveren. */
  int koren_x = sqrt(x);
  for (i = 3; i <= koren_x; i += 2)
    if (x % i == 0)
      return 0;

  /* Ako nijedan od prethodnih uslova nije bio ispunjen, to znaci
     da nijedan broj ne deli x, pa je on prost. */
  return 1;
}

/* Funkcija od niza a formira niz b koji sadrzi sve elemente niza a
   koji nisu prosti brojevi. Povratna vrednost funkcije je broj
   elemenata niza b. */
int obrisi_proste(int a[], int n, int b[]) {
  int i, j;

  /* Kada se u nizu a naidje na prost element, on se upisuje u niz
     b i Uvecavanje brojac za niz b. */
  for (i = 0, j = 0; i < n; i++)
    if (prost(a[i]) == 0) {
```

```
      b[j] = a[i];
      j++;
    }

  return j;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS], b[MAKS];
  int n_a, n_b;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite dimenziju niza: ");
  scanf("%d", &n_a);
  if (n_a <= 0 || n_a > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  ucitaj(a, n_a);

  /* Formira se niz b brisanjem prostih brojeva iz niza a. */
  n_b = obrisi_proste(a, n_a, b);

  /* Ispis elemenata niza b. */
  printf("Rezultujuci niz:\n");
  ispisi(b, n_b);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.31

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 100

/* Funkcija ucitava elemente niza dimenzije n. */
void ucitaj(int a[], int n) {
  int i;
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);
}

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispisi(int niz[], int n) {
  int i;
```

```
17    for (i = 0; i < n; i++)
        printf("%d ", niz[i]);
19    printf("\n");
  }
21
  /* Funkcija brise sve neparne elemente niza. */
23 int obrisi_neparne(int a[], int n) {
    int i, j;
25    /* Promenljiva j predstavlja brojac prve slobodne pozicije na
         koju se moze upisati element niza koji treba da ostane u nizu.
27       Kada se naidje na element koji je paran, on se kopira na
         mesto a[j] i poveca se vrednost brojaca j. Ukoliko se naidje
29       na element koji je neparan, njega treba preskociti. */
    for (i = 0, j = 0; i < n; i++) {
31      /* Ako je tekuci element niza a paran. */
        if (a[i] % 2 == 0) {
33        /* Premesta se na poziciju j. */
          a[j] = a[i];
35
          /* Vrednost brojaca j se priprema za narednu iteraciju. */
37        j++;
        }
39      /* Ako je tekuci element niza a neparan, sa njim nista ne treba
           raditi. */
41    }

43    /* Rezultujuci niz ima j elemenata. */
    return j;
45 }

47 int main() {
    /* Deklaracija potrebnih promenljivih. */
49    int a[MAKS];
    int n;
51
    /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
53    printf("Unesite dimenziju niza: ");
    scanf("%d", &n);
55    if (n <= 0 || n > MAKS) {
        printf("Greska: neispravan unos.\n");
57      exit(EXIT_FAILURE);
    }
59
    /* Ucitavanje elemenata niza. */
61    ucitaj(a, n);

63    /* Brisanje neparnih elemenata niza. */
    n = obrisi_neparne(a, n);
65
    /* Ispis elemenata izmenjenog niza a. */
67    printf("Rezultujuci niz:\n");
    ispisi(a, n);
```

```
69
  exit(EXIT_SUCCESS);
71 }
```

Rešenje 2.1.32 Pogledajte zadatke 2.1.30 i 2.1.31.

Rešenje 2.1.33

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS 700
5
   /* Funkcija ucitava elemente niza dimenzije n. */
7  void ucitaj(int a[], int n) {
     int i;
9    printf("Unesite elemente niza: ");
     for (i = 0; i < n; i++)
11     scanf("%d", &a[i]);
   }
13
   /* Funkcija ispisuje elemente niza dimenzije n. */
15 void ispisi(int a[], int n) {
     int i;
17   for (i = 0; i < n; i++)
       printf("%d ", a[i]);
19   printf("\n");
   }
21
   /* Funkcija pomera za jedno mesto u levo elemente niza a pocevsi od
23    pozicije j. Element na poziciji j se brise i na njegovo mesto se
      upisuje element na poziciji j+1, a u skladu sa tim svi ostali
25    elementi posle njega u nizu se pomeraju. */
   void pomeri_za_jedno_mesto(int a[], int n, int j) {
27   int i;
     for (i = j; i < n - 1; i++)
29     a[i] = a[i + 1];
   }
31
   /* Funkcija brise sve elemente niza koji nisu deljivi svojim
33    indeksom. Povratna vrednost funkcije je broj elemenata
      rezultujuceg niza. */
35 int brisanje(int niz[], int n) {
     int i;
37
     /* Potrebno je krenuti od poslednjeg elementa niza i petljom ici
39      ka pocetku niza (element na poziciji 0 se ne razmatra).
        Proverava se da li je element potrebno obrisati i ako jeste
41      vrsi se pomeranje elemenata niza za jedno mesto u levo.
        Prednost ovog resenja u odnosu na resenje kada se krene od
43      pocetka niza je u tome sto element koji se ispituje sigurno
```

```
     nije promenio svoju poziciju usled pomeranja zbog brisanja.
45   Problem se moze resiti i koriscenjem pomocnog niza (uraditi za
     vezbu). To resenje je efikasnije, ali trosi vise resursa. */
47   for (i = n - 1; i > 0; i--)
       if (niz[i] % i != 0) {
49       pomeri_za_jedno_mesto(niz, n, i);
         /* Nakon brisanja elementa, smanjuje se i dimenzija niza. */
51       n--;
       }
53
   return n;
55 }

57 int main() {
   /* Deklaracija potrebnih promenljivih. */
59 int n, niz[MAKS];

61 /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
   printf("Unesite dimenziju niza: ");
63 scanf("%d", &n);
   if (n <= 0 || n > MAKS) {
65   printf("Greska: neispravan unos.\n");
     exit(EXIT_FAILURE);
67 }

69 /* Ucitavanje elemenata niza. */
   ucitaj(niz, n);
71
   /* Brisanje trazenih elemenata. */
73 n = brisanje(niz, n);

75 /* Ispis rezultujuceg niza. */
   printf("Rezultujuci niz:\n");
77 ispisi(niz, n);

79 exit(EXIT_SUCCESS);
   }
```

Rešenje 2.1.34

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS 500
5
   /* Funkcija vraca 1 ukoliko broj x postoji u nizu, 0 inace. */
7  int postoji(int niz[], int n, int x) {
     int i;
9    for (i = 0; i < n; i++)
       if (niz[i] == x)
11       return 1;
```

```
13    return 0;
  }
15
  /* Funkcija ucitava elemente niza dimenzije n. */
17 void ucitaj(int niz[], int n) {
    int i, element;
19    printf("Unesite elemente niza: ");
    for (i = 0; i < n; i++) {
21      scanf("%d", &element);
      if (postoji(niz, i, element)) {
23        printf("Greska: skup ne moze imati duplikate.\n");
        exit(EXIT_FAILURE);
25      }
      niz[i] = element;
27    }
  }
29
  /* Funkcija ispisuje elemente niza dimenzije n. */
31 void ispisi(int niz[], int n) {
    int i;
33    for (i = 0; i < n; i++)
      printf("%d ", niz[i]);
35    printf("\n");
  }
37
  int main() {
39    /* Deklaracija potrebnih promenljivih. */
    int a[MAKS], b[MAKS], unija[2 * MAKS], presek[MAKS],
41        razlika[MAKS];
    int i, n_a, n_b, n_unija, n_presek, n_razlika;
43
    /* Ucitavanje dimenzije prvog niza i provera ispravnosti ulaza. */
45    printf("Unesite dimenziju niza: ");
    scanf("%d", &n_a);
47    if (n_a <= 0 || n_a > MAKS) {
      printf("Greska: neispravan unos.\n");
49      exit(EXIT_FAILURE);
    }
51
    /* Ucitavanje elemenata niza. */
53    ucitaj(a, n_a);

55    /* Ucitavanje dimenzije drugog niza i provera ispravnosti
       ulaza. */
57    printf("Unesite dimenziju niza: ");
    scanf("%d", &n_b);
59    if (n_b <= 0 || n_b > MAKS) {
      printf("Greska: neispravan unos.\n");
61      exit(EXIT_FAILURE);
    }
63
```

```
  /* Ucitavanje elemenata niza. */
  ucitaj(b, n_b);

  /* Brojaci elemenata u nizovima unija, presek i razlika. */
  n_unija = n_presek = n_razlika = 0;

  for (i = 0; i < n_a; i++) {
    /* Svi elementi niza a se dodaju u uniju. */
    unija[n_unija] = a[i];
    n_unija++;

    /* Ukoliko se element a[i] nalazi u nizu b dodaje se nizu presek
    i
      povecava se brojac elemenata u nizu presek. */
    if (postoji(b, n_b, a[i]) == 1) {
      presek[n_presek] = a[i];
      n_presek++;
    }

    /* Ukoliko element a[i] ne postoji u nizu b dodaje se nizu
    razlika i
      povecava se brojac elemenata u nizu razlika. */
    if (postoji(b, n_b, a[i]) == 0) {
      razlika[n_razlika] = a[i];
      n_razlika++;
    }
  }

  /* Elemente niza b koji nisu uneti u uniju dodaju se u uniju. */
  for (i = 0; i < n_b; i++)
    if (postoji(unija, n_unija, b[i]) == 0) {
      unija[n_unija] = b[i];
      n_unija++;
    }

  /* Ispis rezultata. */
  printf("Unija: ");
  ispisi(unija, n_unija);

  printf("Presek: ");
  ispisi(presek, n_presek);

  printf("Razlika: ");
  ispisi(razlika, n_razlika);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.1.35

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 2000

/* Funkcija ispisuje elemente niza dimenzije n. */
void ispis(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", niz[i]);
  printf("\n");
}

/* Funkcija ubacuje element x na kraj niza i vraca novu dimenziju
   niza. */
int ubaci_na_kraj(int niz[], int n, int x) {
  if (n == MAKS) {
    printf("Greska: prekoracen je maksimalan broj elemenata niza.");
    exit(EXIT_FAILURE);
  }

  niz[n] = x;
  return n + 1;
}

/* Funkcija ubacuje element x na pocetak niza i vraca novu dimenziju
   niza. */
int ubaci_na_pocetak(int niz[], int n, int x) {
  if (n == MAKS) {
    printf("Greska: prekoracen je maksimalan broj elemenata niza.");
    exit(EXIT_FAILURE);
  }

  int i;
  /* Prvo se svi elementi niza pomere za jednu poziciju u desno da
     bi se oslobodio prostor za prvi element niza. Poslednji
     element niza se pomera sa pozicije (n-1) na poziciju (n).
     Slicno se pomeraju i ostali elementi. */
  for (i = n; i > 0; i--)
    niz[i] = niz[i - 1];

  /* Na prvu poziciju se upisuje novi element. Bitan je redosled
     naredbi: ako bi prvo bio upisan novi element, a tek onda
     izvrseno pomeranje, element na poziciji niz[0] bi bio obrisan
     i ne bi mogao biti upisan na poziciju niz[1]. */
  niz[0] = x;

  return n + 1;
}
```

```
/* Funkcija ubacuje element x na neku poziciju u nizu i vraca novu
   dimenziju niza. */
int ubaci_na_poziciju(int niz[], int n, int x, int pozicija) {
  if (n == MAKS) {
    printf("Greska: prekoracen je maksimalan broj elemenata niza.");
    exit(EXIT_FAILURE);
  }

  int i;
  /* Prvo se svi elementi niza od pozicije do kraja pomere za jedno
     mesto u desno da bi se oslobodio prostor za novi element niza.
     */
  for (i = n; i > pozicija; i--)
    niz[i] = niz[i - 1];

  /* Na poziciju se upisuje novi element. */
  niz[pozicija] = x;

  return n + 1;
}

/* Funkcija brise prvi element niza i vraca novu dimenziju niza. */
int brisi_prvog(int niz[], int n) {
  if (n == 0) {
    printf("Greska: nije moguce brisanje iz praznog niza.\n");
    exit(EXIT_FAILURE);
  }

  int i;
  /* Svi elementi niza pomeraju se za jedno mesto u levo. */
  for (i = 0; i < n - 1; i++)
    niz[i] = niz[i + 1];

  return n - 1;
}

/* Funkcija brise poslednji element niza i vraca novu dimenziju
   niza. */
int brisi_poslednjeg(int niz[], int n) {
  if (n == 0) {
    printf("Greska: nije moguce brisanje iz praznog niza.\n");
    exit(EXIT_FAILURE);
  }

  /* Dovoljno je smanjiti dimenziju niza, elemente niza nije
     potrebno brisati. */
  return n - 1;
}

/* Funkcija brise element x i vraca novu dimenziju niza.
   Pretpostavlja se da element ima samo jedno pojavljivanje. */
int brisi_element(int niz[], int n, int x) {
```

```
  int i, j;

  /* Prvo treba pronaci poziciju elementa u nizu. */
  for (i = 0; i < n; i++)
    if (niz[i] == x)
      break;

  /* Provera da li element postoji u nizu. Ako je brojac stigao do
     kraja niza, onda element ne postoji u nizu. */
  if (i == n) {
    printf("Klijent sa rednim brojem %d ne postoji u nizu.\n", x);
    return n;
  }

  /* Ukoliko element postoji u nizu, svi elementi niza nakon njega
     se pomeraju za jedno mesto u levo. */
  for (j = i; j < n - 1; j++)
    niz[j] = niz[j + 1];

  return n - 1;
}

int main() {
  int n, niz[MAKS], i, klijent, pozicija;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite trenutni broj klijenata: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  printf("Unesite niz sa rednim brojevima klijenata: ");
  for (i = 0; i < n; i++)
    scanf("%d", &niz[i]);

  /* Ubacivanje klijenta na kraj. */
  printf("Unesite broj klijenta kojeg treba ubaciti u niz: ");
  scanf("%d", &klijent);
  n = ubaci_na_kraj(niz, n, klijent);
  printf("Niz nakon ubacivanja klijenta:\n");
  ispis(niz, n);

  /* Ubacivanje klijenta na pocetak. */
  printf("Unesite prioritetnog klijenta kojeg treba"
         "ubaciti u niz: ");
  scanf("%d", &klijent);
  n = ubaci_na_pocetak(niz, n, klijent);
  printf("Niz nakon ubacivanja klijenta:\n");
  ispis(niz, n);
```

```
155   /* Ubacivanje klijenta na zadatu poziciju. */
      printf("Unesite prioritetnog klijenta kojeg treba ubaciti "
157          "u niz i njegovu poziciju:");
      scanf("%d%d", &klijent, &pozicija);
159   if (pozicija < 0 || pozicija > n) {
        printf("Greska: neispravan unos.\n");
161     exit(EXIT_FAILURE);
      } else {
163     n = ubaci_na_poziciju(niz, n, klijent, pozicija);
        printf("Niz nakon ubacivanja klijenta:\n");
165     ispis(niz, n);
      }
167
      /* Brisanje prvog klijenta. */
169   n = brisi_prvog(niz, n);
      printf("Niz nakon odlaska klijenta:\n");
171   ispis(niz, n);

173   /* Brisanje poslednjeg klijenta. */
      n = brisi_poslednjeg(niz, n);
175   printf("Niz nakon odlaska klijenta:\n");
      ispis(niz, n);
177
      /* Brisanje klijenta sa datim rednim brojem. */
179   printf("Unesite redni broj klijenta koji je napustio red: ");
      scanf("%d", &klijent);
181   n = brisi_element(niz, n, klijent);
      printf("Niz nakon odlaska klijenta:\n");
183   ispis(niz, n);

185   exit(EXIT_SUCCESS);
    }
```

2.3 Pokazivači

Zadatak 2.3.1 Napisati funkciju `void uredi(int *pa, int *pb)` koja uređuje svoja dva celobrojna argumenta tako da se u prvom nalazi manja vrednost, a u drugom veća. Napisati program koji učitava dva cela broja i ispisuje uređene brojeve.

Primer 1

```
Interakcija sa programom:
 Unesite dva broja: 2 5
 Uredjene promenljive: 2, 5
```

Primer 2

```
Interakcija sa programom:
 Unesite dva broja: 11 -4
 Uredjene promenljive: -4, 11
```

Zadatak 2.3.2 Napisati funkciju `void rgb_u_cmy(int r, int g, int b, float *c, float *m, float *y)` koja datu boju u *rgb* formatu konvertuje u boju u *cmy* formatu po sledećim formulama:

$$c = 1 - r/255, \quad m = 1 - g/255, \quad y = 1 - b/255$$

Napisati program koji učitava boju u *rgb* formatu i ispisuje vrednosti unete boje u *cmy* formatu. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. Napomena: *Vrednosti boja u rgb formatu su u opsegu* $[0, 255]$.

Primer 1

```
Interakcija sa programom:
 Unesite boju u rgb formatu: 56 111 24
 cmy: (0.78, 0.56, 0.91)
```

Primer 2

```
Interakcija sa programom:
 Unesite boju u rgb formatu: 156 -90 5
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite boju u rgb formatu: 9 0 237
 cmy: (0.96, 1.00, 0.07)
```

Primer 4

```
Interakcija sa programom:
 Unesite boju u rgb formatu: 300 11 27
 Greska: neispravan unos.
```

Zadatak 2.3.3 Napisati funkciju `int presek(float k1, float n1, float k2, float n2, float *px, float *py)` koja za dve razne prave date svojim koeficijentima pravca i slobodnim članovima određuje njihovu tačku preseka. Funkcija treba da vrati jedinicu ako se prave seku, a nulu ako nemaju tačku preseka (ako su paralelne). Napisati program koji učitava podatke o pravama i ukoliko prave imaju presek, ispisuje koordinate tačke preseka, a ako nemaju, ispisuje odgovarajuću poruku.

Primer 1

```
Interakcija sa programom:
 Unesite k i n za prvu pravu: 4 5
 Unesite k i n za drugu pravu: 11 -4
 Prave se seku u tacki (1.29, 10.14).
```

Primer 2

```
Interakcija sa programom:
 Unesite k i n za prvu pravu: 0.5 -4.7
 Unesite k i n za drugu pravu: 0.5 9.1
 Prave su paralelne.
```

Zadatak 2.3.4 Napisati funkciju koja za dva cela broja izračunava njihov količnik i ostatak pri deljenju. Funkcija treba da vrati jedinicu ukoliko je uspešno izračunala vrednosti, a nulu ukoliko deljenje nije moguće. Napisati program koji učitava dva cela broja i ispisuje njihov količnik i ostatak pri deljenju. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite brojeve: 4 5
 Kolicnik: 0
 Ostatak: 4
```

Primer 2

```
Interakcija sa programom:
 Unesite brojeve: 4 0
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite brojeve: -123 11
 Kolicnik: -11
 Ostatak: -2
```

Zadatak 2.3.5 Napisati funkciju koja za dužinu trajanja filma koja je data u sekundama, određuje ukupno trajanje filma u satima, minutima i sekundama. Napisati program koji učitava trajanje filma u sekundama i ispisuje odgovarajuće vreme trajanja u formatu *broj_sati*h:*broj_minuta*m:*broj_sekundi*s. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Trajanje fima u sekundama: 5000
 1h:23m:20s
```

Primer 2

```
Interakcija sa programom:
 Trajanje fima u sekundama: -300
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Trajanje fima u sekundama: 2500
 0h:41m:40s
```

Primer 4

```
Interakcija sa programom:
 Trajanje fima u sekundama: 7824
 2h:10m:24s
```

Zadatak 2.3.6 Napisati funkciju koja sa ulaza učitava karakter po karakter sve do kraja ulaza, a zatim prebrojava sva pojavljivanja karaktera tačka i sva pojavljivanja karaktera zarez. Napisati program koji za uneti tekst ispisuje koliko puta se pojavila tačka, a koliko puta se pojavio zarez.

Primer 1

```
Interakcija sa programom:
 Unesite tekst:
 Bio jednom jedan lav...
 Kakav lav?
 Strasan lav,
 narogusen i ljut sav!
 Broj tacaka: 3
 Broj zareza: 1
```

Primer 2

```
Interakcija sa programom:
 Unesite tekst:
 Bavite se sportom,
 ne moze da skodi,
 sportisti su bili
 i bice u modi.
 Kondicije puni,
 uvek vedri, zdravi.
 Svako dete treba
 sportom da se bavi.
 Broj tacaka: 3
 Broj zareza: 4
```

Primer 3

```
Interakcija sa programom:
 Unesite tekst:
 Na sirokom carskom drumu
 sto preseca prasumu
 sreli se beli slon
 i jedan crni telefon!
 Broj tacaka: 0
 Broj zareza: 0
```

Zadatak 2.3.7 Napisati funkciju `void par_nepar(int a[], int n, int parni[], int *np, int neparni[], int *nn)` koja razbija niz a na niz parnih i niz neparnih brojeva. Vrednost na koju pokazuje pokazivač np treba da bude jednaka broju elemenata niza `parni`, a vrednost na koju pokazuje pokazivač nn treba da bude jednaka broju elemenata niza `neparni`. Maksimalan broj elemenata niza je 50. Napisati program koji učitava dimenziju niza, a zatim i elemente niza i ispisuje odgovarajuće nizove parnih, odnosno neparnih elemenata unetog niza. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj elemenata niza: 8
 Unesite elemente niza:
 1 8 9 -7 -16 24 77 4
 Niz parnih brojeva: 8 -16 24 4
 Niz neparnih brojeva: 1 9 -7 77
```

Primer 2

```
Interakcija sa programom:
 Unesite broj elemenata niza: 5
 Unesite elemente niza:
 2 4 6 8 -11
 Niz parnih brojeva: 2 4 6 8
 Niz neparnih brojeva: -11
```

Primer 3

```
Interakcija sa programom:
 Unesite broj elemenata niza: 2
 Unesite elemente niza: -15 15
 Niz parnih brojeva:
 Niz neparnih brojeva: -15 15
```

Primer 4

```
Interakcija sa programom:
 Unesite broj elemenata niza: 0
 Greska: neispravan unos.
```

Zadatak 2.3.8 Napisati funkciju koja izračunava najmanji i najveći element niza realnih brojeva. Napisati program koji učitava niz realnih brojeva maksimalne dužine 50 i ispisuje vrednosti najmanjeg i najvećeg elementa niza, zaokružene na tri decimale. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj elemenata niza: 5
 Unesite elemente niza:
 24.16 -32.11 999.25 14.25 11
 Najmanji: -32.110
 Najveci: 999.250
```

Primer 2

```
Interakcija sa programom:
 Unesite broj elemenata niza: 4
 Unesite elemente niza:
 -5.126 -18.29 44 29.268
 Najmanji: -18.290
 Najveci: 44.000
```

Primer 3

```
Interakcija sa programom:
 Unesite broj elemenata niza: 1
 Unesite elemente niza: 4.16
 Najmanji: 4.160
 Najveci: 4.160
```

Primer 4

```
Interakcija sa programom:
 Unesite broj elemenata niza: -3
 Greska: neispravan unos.
```

2.4 Rešenja

Rešenje 2.3.1

```
#include <stdio.h>
#include <stdlib.h>

/* Argumenti funkcije uredi_pogresno, promenljive a i b,
   predstavljaju lokalne promenljive za ovu funkciju i prestaju da
   postoje po zavrsetku funkcije. Zbog toga se efekti razmene
   vrednosti promenljivih a i b u slucaju da je a>b ne vide u
   glavnom programu.
   void uredi_pogresno(int a, int b) {
     int pom;
     if (a > b) {
       pom = a;
       a = b;
       b = pom;
     }
   } */

/* Argumenti funkcije uredi, promenljive pa i pb, takodje su
   lokalne promenljive za ovu funkciju i prestaju da postoje kada
   se funkcija zavrsi. Razlika je u tome sto su one adrese
   promenljivih a i b koje zelimo da razmenimo u slucaju da je a>b.

   Promenljivoj a se pristupa preko pokazivacke promenljive pa sa
   *pa i slicno, promenljivoj b sa *pb.

   Vrednosti promenljivih *pa i *pb se razmenjuju kao i vrednosti
   bilo koje dve celobrojne promenljive. */
void uredi(int *pa, int *pb) {
  int pom;
  if (*pa > *pb) {
    pom = *pa;
    *pa = *pb;
    *pb = pom;
  }
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int a, b;

  /* Ucitavanje vrednosti dva cela broja. */
  printf("Unesite dva broja:");
  scanf("%d%d", &a, &b);

  /* Neispravan nacin:
     uredi_pogresno(a, b);
     printf("Uredjene promenljive: %d, %d\n", a, b); */
```

```
49   /* Funkcija uredi kao argumente prima dve pokazivacke promenljive
        (int*,int*). Zbog toga je u pozivu funkcije neophodno
51      proslediti adrese promenljivih koje zelimo da uredimo rastuce:
        &a i &b. */
53   uredi(&a, &b);
     printf("Uredjene promenljive: %d, %d\n", a, b);
55
     exit(EXIT_SUCCESS);
57 }
```

Rešenje 2.3.2

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MIN_RGB 0
5  #define MAKS_RGB 255

7  /* Funkcija vrsi konverziju boje iz rgb formata u cmy format.
      Kako se pomocu naredbe return ne moze vratiti vise od jedne
9     vrednosti, neophodno je da se promenljive cije se vrednosti
      racunaju prenesu preko pokazivaca. */
11 void rgb_u_cmy(int r, int g, int b, float *c, float *m, float *y) {
     *c = 1 - r / 255.0;
13   *m = 1 - g / 255.0;
     *y = 1 - b / 255.0;
15 }

17 /* Funkcija proverava da li je vrednost boje u ispravnom opsegu. */
   int ispravna_rgb_vrednost(int boja) {
19   if (boja < MIN_RGB || boja > MAKS_RGB)
       return 0;
21   return 1;
   }
23
   int main() {
25   /* Deklaracije potrebnih promenljivih. */
     int r, g, b;
27   float c, m, y;

29   /* Ucitavanje vrednosti boje u rgb formatu. */
     printf("Unesite boju u rgb formatu: ");
31   scanf("%d%d%d", &r, &g, &b);

33   /* Provera ispravnosti ulaza. */
     if (!ispravna_rgb_vrednost(r) || !ispravna_rgb_vrednost(g) ||
35       !ispravna_rgb_vrednost(b)) {
       printf("Greska: neispravan unos.\n");
37     exit(EXIT_FAILURE);
     }
```

```
39
   /* Konverzija boje i ispis rezultata. Funkciji se kao argumenti
41    prosledjuju vrednosti brojeva r, g, i b, kao i adrese na koje
      treba da se upisu izracunate c, m, y vrednosti. */
43 rgb_u_cmy(r, g, b, &c, &m, &y);
   printf("cmy: (%.2f, %.2f, %.2f)\n", c, m, y);
45
   exit(EXIT_SUCCESS);
47 }
```

Rešenje 2.3.3

```
   #include <stdio.h>
2  #include <stdlib.h>

4  /* Funkcija racuna presek pravih
      y = k1 * x + n1 i y = k2 * x + n2.
6     Koordinate preseka (ako postoji) se upisuju na adrese px i
      py. Kao povratna vrednost funkcije se vraca jedinica ukoliko
8     presek postoji, a nula inace. */
   int presek(float k1, float n1, float k2, float n2, float *px,
10            float *py) {
     /* Ako je koeficijent pravca jednak, prave su paralelne. */
12   if (k1 == k2)
       return 0;
14
     /* Koordinate preseka se upisuju na adrese (px, py). */
16   *px = -(n1 - n2) / (k1 - k2);
     *py = k1 * (*px) + n1;
18
     /* Funkcija vraca 1 kao indikator da presek postoji. */
20   return 1;
   }
22
   int main() {
24   /* Deklaracije potrebnih promenljivih. */
     float k1, k2, n1, n2;
26   float x, y;

28   /* Ucitavanje parametara za dve prave. */
     printf("Unesite k i n za prvu pravu: ");
30   scanf("%f%f", &k1, &n1);
     printf("Unesite k i n za drugu pravu: ");
32   scanf("%f%f", &k2, &n2);

34   /* Ispis rezultata. */
     if (presek(k1, n1, k2, n2, &x, &y))
36     printf("Prave se seku u tacki (%.2f, %.2f).\n", x, y);
     else
38     printf("Prave su paralelne.\n");
```

```
  exit(EXIT_SUCCESS);
}
```

Rešenje 2.3.4 Pogledajte zadatke 2.3.2 i 2.3.3.

Rešenje 2.3.5

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija koja dato trajanje izrazeno u ukupnom broju sekundi
   konvertuje u trajanje koje je izrazeno u broju sati, minuta i
   sekundi. */
void konverzija(int trajanje, int *psati, int *pminuti,
                int *psekunde) {
  *psati = trajanje / 3600;
  trajanje -= *psati * 3600;

  *pminuti = trajanje / 60;
  trajanje -= *pminuti * 60;

  *psekunde = trajanje;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int trajanje, sati, minuti, sekunde;

  /* Ucitavanje trajanja u sekundama i provera ispravnosti ulaza. */
  printf("Trajanje filma u sekundama: ");
  scanf("%d", &trajanje);
  if (trajanje < 0) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Racunanje i ispis rezultata. */
  konverzija(trajanje, &sati, &minuti, &sekunde);
  printf("%dh:%dm:%ds\n", sati, minuti, sekunde);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.3.6

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija ucitava karakter po karakter sa ulaza i prebrojava
   koliko puta se pojavio karakter '.' i koliko puta se pojavio
```

```
   karakter ','. */
void interpunkcija(int *br_tacaka, int *br_zareza) {
  int tacke = 0, zarezi = 0;
  char c;

  /* Ucitavanje i prebrojavanje trazenih karaktera. */
  while ((c = getchar()) != EOF) {
    if (c == '.')
      tacke++;

    if (c == ',')
      zarezi++;
  }

  /* Smestanje rezultata na prosledjene adrese. */
  *br_tacaka = tacke;
  *br_zareza = zarezi;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int br_tacaka, br_zareza;

  /* Ucitavanje i obrada teksta. */
  printf("Unesite tekst: \n");
  interpunkcija(&br_tacaka, &br_zareza);

  /* Ispis rezultata. */
  printf("Broj tacaka: %d\n", br_tacaka);
  printf("Broj zareza: %d\n", br_zareza);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.3.7

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija od niza a formira dva niza: niz parnih elemenata
   niza a i niz neparnih elemenata niza a. Duzine rezultujucih
   nizova se upisuju na adrese np i nn. */
void par_nepar(int a[], int n, int parni[], int *np,
               int neparni[], int *nn) {
  int i, j, k;

  /* Promenljiva i je brojac u originalnom nizu i on se uvecava u
     svakoj iteraciji. Promenljiva j je projac za niz parnih
     brojeva i on treba da se uveca svaki put kada se naidje na
```

```
     novi element ovog niza. Promenljiva k je brojac za niz
     neparnih brojeva i on treba da se uveca sveki put kada se
     naidje na novi element ovog niza. */
  for (i = 0, j = 0, k = 0; i < n; i++) {
    if (a[i] % 2 == 0) {
      parni[j] = a[i];
      j++;
    } else {
      neparni[k] = a[i];
      k++;
    }
  }

  /* Na kraju petlje, u promenljivoj j se nalazi podatak o broju
     elementa niza parni[], a u promenljivoj k podatak o broju
     elementa niza neparni[]. Ove vrednosti se upisuju na adrese np
     i nn. */
  *np = j;
  *nn = k;
}

/* Funkcija ispisuje elemente niza. */
void ispisi(int niz[], int n) {
  int i;
  for (i = 0; i < n; i++)
    printf("%d ", niz[i]);
  printf("\n");
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, n1, n2, i;
  int a[MAKS], parni[MAKS], neparni[MAKS];

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite broj elemenata niza: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata niza. */
  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &a[i]);

  /* Popunjavanje rezultujucih nizova odgovarajucim
     vrednostima. */
  par_nepar(a, n, parni, &n1, neparni, &n2);

  /* Ispis niza parni[] koji ima n1 elemenata. */
```

```
  printf("Niz parnih brojeva: ");
  ispisi(parni, n1);

  /* Ispis niza neparni[] koji ima n2 elemenata. */
  printf("Niz neparnih brojeva: ");
  ispisi(neparni, n2);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.3.8

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija racuna najmanji i najveci element niza a duzine n. */
void min_maks(float a[], int n, float *najmanji, float *najveci) {
  int i;

  /* Vrednosti minimuma i maksimuma se inicijalizuju na vrednost
     prvog clana niza. */
  *najmanji = a[0];
  *najveci = a[0];

  /* U petlji se prolazi kroz ostale clanove niza i po potrebi se
     vrsi azuriranje najmanje i najvece vrednosti. */
  for (i = 1; i < n; i++) {
    if (a[i] > *najveci)
      *najveci = a[i];

    if (a[i] < *najmanji)
      *najmanji = a[i];
  }

  /* Na kraju petlje, na adresama najmanji i najveci se nalaze
     trazene vrednosti. */
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int i, n;
  float a[MAKS], min, maks;

  /* Ucitavanje dimenzije niza i provera ispravnosti ulaza. */
  printf("Unesite broj elemenata niza: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
```

```
  }

  /* Ucitavanje elemenata niza. */
  printf("Unesite elemente niza:\n");
  for (i = 0; i < n; i++)
    scanf("%f", &a[i]);

  /* Racunanje vrednosti najmanjeg i najveceg elementa. */
  min_maks(a, n, &min, &maks);

  /* Ispis rezultata. */
  printf("Najmanji: %.3f\n", min);
  printf("Najveci: %.3f\n", maks);

  exit(EXIT_SUCCESS);
}
```

2.5 Niske

Zadatak 2.5.1 Napisati funkciju `void konvertuj(char s[])` koja menja nisku s tako što mala slova zamenjuje odgovarajućim velikim slovima, a velika slova zamenjuje odgovarajućim malim slovima. Napisati program koji učitava nisku maksimalne dužine 10 karaktera i ispisuje konvertovanu nisku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: BeoGrad
 Konvertovana niska: bEOgRAD
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: A+B+C
 Konvertovana niska: a+b+c
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 12345
 Konvertovana niska: 12345
```

Zadatak 2.5.2 Napisati funkciju `void ubaci_zvezdice(char s[])` koja menja nisku s tako što u njoj svaki drugi karakter zamenjuje zvezdicom. Napisati program koji učitava nisku maksimalne dužine 20 karaktera i ispisuje izmenjenu nisku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: *a*b*c*
 Izmenjena niska: *******
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: zimA
 Izmenjena niska: z*m*
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 123abc789
 Izmenjena niska: 1*3*b*7*9
```

Zadatak 2.5.3 Napisati program koji vrši poređenje niski. Napisati funkcije:

(a) `int jednake(char s1[], char s2[])` koja vraća jedinicu ako su s_1 i s_2 jednake niske, a nulu inače.

(b) `void u_velika_slova(char s[])` koja pretvara sva slova niske s u velika slova, a ostale karaktere ne menja.

Program učitava dve reči maksimalne dužine 20 karaktera i ispituje da li su unete reči jednake. Pri poređenju treba zanemariti razliku između malih i velikih slova.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite niske:
 isPit2010
 IsPiT2010
 Niske su jednake.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite niske:
 Prog1
 prog2
 Niske nisu jednake.
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite niske:
 jun
 JUNSKI
 Niske nisu jednake.
```

Zadatak 2.5.4 Napisati program koji proverava da li se uneta niska završava samoglasnikom. Napisati funkcije:

(a) `int samoglasnik(char c)` koja ispituje da li je karakter c samoglasnik i vraća 1 ako jeste ili 0 ako nije.

(b) `int samoglasnik_na_kraju(char s[])` koja ispituje da li se niska s završava samoglasnikom.

Program učitava reč maksimalne dužine 20 karaktera i ispisuje da li se reč završava samoglasnikom ili ne.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: kestenje
 Niska se zavrsava samoglasnikom.
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: vetar
 Niska se ne zavrsava samoglasnikom.
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: OLUJA
 Niska se zavrsava samoglasnikom.
```

Primer 4

```
Interakcija sa programom:
 Unesite nisku: Programiranje1
 Niska se ne zavrsava samoglasnikom.
```

Zadatak 2.5.5 Napisati funkciju `int sadrzi_veliko(char s[])` koja proverava da li niska s sadrži veliko slovo. Napisati program koji za učitanu nisku maksimalne dužine 20 karaktera proverava da li sadrži veliko slovo i ispisuje odgovarajuću poruku.

Primer 1

```
Interakcija sa programom:
 Unesite nisku:
 naocare
 Ne sadrzi veliko slovo.
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku:
 DiopTrija0.75
 Sadrzi veliko slovo.
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku:
 21.06.2017.
 Ne sadrzi veliko slovo.
```

Zadatak 2.5.6 Napisati program koji za učitanu nisku s i karakter c ispituje da li se karakter c pojavljuje u niski s. Ako je to slučaj, program treba da ispiše indeks prvog pojavljivanja karaktera c u niski s, a u suprotnom -1. Pretpostaviti da niska može da ima najviše 20 karaktera.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: bazen
 Unesite karakter: z
 Pozicija: 2
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: lezaljka
 Unesite karakter: a
 Pozicija: 3
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: limunada
 Unesite karakter: b
 Pozicija: -1
```

Zadatak 2.5.7 Napisati funkciju `int podniska(char s[], char t[])` koja proverava da li je niska t uzastopna podniska niske s. Napisati program koji učitava dve niske maksimalne dužine 10 karaktera i ispisuje da li je druga niska

podniska prve.

Primer 1

```
Interakcija sa programom:
 Unesite nisku s: abcde
 Unesite nisku t: bcd
 t je podniska niske s.
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku s: abcde
 Unesite nisku t: bCd
 t nije podniska niske s.
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku s: abcde
 Unesite nisku t: def
 t nije podniska niske s.
```

Zadatak 2.5.8 Napisati funkciju `void skrati(char s[])` koja uklanja beline sa kraja date niske. Napisati program koji učitava liniju maksimalne dužine 100 karaktera i ispisuje učitanu i izmenjenu nisku između zvezdica.

Primer 1

```
Interakcija sa programom:
 Unesite nisku:
 rep belina
 Ucitana niska:
 *rep belina                *
 Izmenjena niska:
 *rep belina*
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku:
 tri tabulatora na kraju
 Ucitana niska:
 *tri tabulatora na kraju         *
 Izmenjena niska:
 *tri tabulatora na kraju*
```

Zadatak 2.5.9 Napisati funkciju `void ukloni_slova(char s[])` koja iz niske s uklanja sva mala i sva velika slova. Napisati program koji za učitanu nisku maksimalne dužine 20 karaktera ispisuje odgovarajuću izmenjenu nisku.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: a1b2c3def
 Rezultat: 123
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: 1+2=3
 Rezultat: 1+2=3
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: malaVELIKA
 Rezultat:
```

Zadatak 2.5.10 Napisati funkciju `void ukloni(char *s)` koja iz niske uklanja sva slova iza kojih sledi slovo koje je u engelskoj abecedi nakon njih, pri čemu se veličina slova zanemaruje. Pravilo se ne primenjuje na nisku dobijenu uklanjanjem. Napisati program koji učitava liniju teksta koja ima najviše 100 karaktera i ispisuje liniju koja se dobije nakon uklanjanja pomenutih karaktera.

Primer 1

```
Interakcija sa programom:
 Unesite nisku:
 Zdravo svima!
 Izmenjena niska:
 Zrvo vma!
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku:
 Danas je 10 stepeni.
 Izmenjena niska:
 Dns j 10 tpni.
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku:
 Ima vetra, kise i hladnoce.
 Izmenjena niska:
 ma vtra, kse i loe.
```

Zadatak 2.5.11 Napisati program koji učitava nisku s maksimalne dužine 30 karaktera i formira nisku t trostrukim nadovezivanjem niske s.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: dan
 Rezultujuca niska:
 dandandan
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: 3sesira
 Rezultujuca niska:
 3sesira3sesira3sesira
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: a-b=5
 Rezultujuca niska:
 a-b=5a-b=5a-b=5
```

Zadatak 2.5.12 Napisati program koji za unetu reč maksimalne dužine 20 karaktera i pozitivan broj n manji od 10, formira rezultujuću reč tako što unetu reč kopira n puta pri čemu se između svaka dva kopiranja umeće crtica. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: ana
 Unesite broj n: 4
 Rezultujuca niska:
 ana-ana-ana-ana
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: 123
 Unesite broj n: 1
 Rezultujuca niska:
 123
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: x*y
 Unesite broj n: 3
 Rezultujuca niska:
 x*y-x*y-x*y
```

Zadatak 2.5.13 Napisati funkciju `void kopiraj_n(char t[], char s[], int n)` koja kopira najviše n karaktera niske s u nisku t. Napisati program koji testira rad napisane funkcije. Pretpostaviti da je maksimalna dužina niske s 20 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: petar
 Unesite broj n: 3
 Rezultujuca niska: pet
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: gromobran
 Unesite broj n: 4
 Rezultujuca niska: grom
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: abc
 Unesite broj n: 15
 Rezultujuca niska: abc
```

Zadatak 2.5.14 Napisati funkciju `void dupliranje(char t[], char s[])` koja na osnovu niske s formira nisku t tako što duplira svaki karakter niske s. Napisati program koji učitava nisku maksimalne dužine 20 karaktera i ispisuje nisku koja se dobije nakon dupliranja karaktera.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: zima
 Rezultujuca niska: zziimmaa
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: C++
 Rezultujuca niska: CC++++
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: C
 Rezultujuca niska: CC
```

Zadatak 2.5.15 Napisati program koji učitava nisku cifara sa eventualnim

vodećim znakom i pretvara je u ceo broj. NAPOMENA: *Pretpostaviti da je unos ispravan.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: -1238
 Rezultat: -1238
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 73
 Rezultat: 73
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: +1
 Rezultat: 1
```

Zadatak 2.5.16 Napisati program koji učitava ceo broj, pretvara ga u nisku i ispisuje dobijenu nisku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite ceo broj: -6543
 Rezultat: -6543
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite ceo broj: 84
 Rezultat: 84
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite ceo broj: 5
 Rezultat: 5
```

Zadatak 2.5.17 Napisati funkciju `int heksadekadni_broj(char s[])` koja proverava da li je niskom s zadat korektan heksadekadni broj. Funkcija treba da vrati vrednost 1 ukoliko je uslov ispunjen, odnosno 0 ako nije. Napisati program koji za učitanu nisku maksimalne dužine 7 karaktera ispisuje da li je korektan heksadekadni broj. UPUTSTVO: *Heksadekadni broj je korektno zadat ako počinje prefiksom* $0x$ *ili* $0X$ *i ako sadrži samo cifre i mala ili velika slova* A*,* B*,* C*,* D*,* E *i* F*.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0x12EF
 Korektan heksadekadni broj.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0X22af
 Korektan heksadekadni broj.
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0xErA9
 Nekorektan heksadekadni broj.
```

Zadatak 2.5.18 Napisati funkciju `int dekadna_vrednost(char s[])` koja izračunava dekadnu vrednost heksadekadnog broja zadatog niskom s. Napisati program koji za učitanu nisku maksimalne dužine 7 karaktera ispisuje odgovarajuću dekadnu vrednost. Pretpostaviti da je uneta niska korektan heksadekadni broj.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0x2A34
 Rezultat: 10804
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0Xff2
 Rezultat: 4082
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: 0xE1A9
 Rezultat: 57769
```

Zadatak 2.5.19 Napisati funkciju `int ucitaj_liniju(char s[], int n)` koja učitava liniju maksimalne dužine n u nisku s i vraća dužinu učitane linije. Napisati program koji učitava linije do kraja ulaza i ispisuje najdužu liniju i njenu dužinu. Ukoliko ima više linija maksimalne dužine, ispisati prvu. Pretpostaviti da svaka linija sadrži najviše 80 karaktera. NAPOMENA: *Linija može da sadrži blanko znakove, ali ne može sadržati znak za novi red ili* `EOF`.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite tekst:
 Dobar dan!
 Kako ste, sta ima novo?
 Ja sam dobro.
 Najduza linija:
 Kako ste, sta ima novo?
 Duzina: 23
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite tekst:
 Prva linija
 Druga linija
 Treca linija
 Najduza linija:
 Druga linija
 Duzina: 12
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite tekst:
 Danas je lep dan.
 Najduza linija:
 Danas je lep dan.
 Duzina: 17
```

*** Zadatak 2.5.20** Napisati funkcije za rad sa rečenicama:

(a) `int procitaj_recenicu(char s[], int n)` koja učitava rečenicu sa ulaza i smešta je u nisku s. Funkcija vraća dužinu učitane rečenice. Učitavanje se završava nakon učitanog karaktera `.`, nakon n učitanih karaktera ili ako se dođe do kraja ulaza.

(b) `void prebroj(char s[], int *broj_malih, int *broj_velikih)` koja prebrojava mala i velika slova u niski s.

Napisati program koji učitava rečenice do kraja ulaza i ispisuje onu rečenicu kod koje je apsolutna razlika broja malih i velikih slova najveća. Pri učitavanju rečenica zanemariti sve beline koje se nalaze između dve rečenice. Pretpostaviti da jedna rečenica sadrži najviše 80 karaktera.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite tekst:
 U ovom poglavlju se govori o niskama. Niske su nizovi karaktera ciji je
 poslednji element terminalna nula.
 U ovom zadatku je potrebno ucitati recenice. Svaka recenica pocinje sa bilo
 kojim karakterom koji nije belina. Na kraju recenice se nalazi tacka.
 Rezultujuca recenica:
 Niske su nizovi karaktera ciji je poslednji element terminalna nula.
```

Zadatak 2.5.21 Napisati funkciju `char* strchr_klon(char s[], char c)` koja vraća pokazivač na prvo pojavljivanje karaktera c u niski s ili $NULL$ ukoliko se karakter c ne pojavljuje u niski s. [1] Napisati program koji za učitanu nisku

[1] Funkcija *strchr_klon* odgovara funkciji *strchr* čija se deklaracija nalazi u zaglavlju string.h.

maksimalne dužine 20 karaktera i karakter c ispisuje indeks prvog pojavljivanja karaktera c u okviru učitane niske ili -1 ukoliko učitana niska ne sadrži uneti karakter.

Primer 1

```
Interakcija sa programom:
 Unesite nisku s: programiranje
 Unesite karakter c: a
 Pozicija: 5
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku s: 123456789
 Unesite karakter c: y
 Pozicija: -1
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku s: leto2017
 Unesite karakter c: 0
 Pozicija: 5
```

Primer 4

```
Interakcija sa programom:
 Unesite nisku s: jedrilica
 Unesite karakter c: I
 Pozicija: -1
```

Zadatak 2.5.22 Napisati funkciju `int strspn_klon(char t[], char s[])` koja izračunava dužinu prefiksa niske t sastavljenog od karaktera niske s. Napisati program koji za učitane dve niske maksimalne dužine 20 karaktera ispisuje rezultat poziva napisane funkcije.

Primer 1

```
Interakcija sa programom:
 Unesite nisku t: program
 Unesite nisku s: pero
 Rezultat: 3
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku t: Barselona
 Unesite nisku s: Brazil
 Rezultat: 3
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku t: 24.10.2017.
 Unesite nisku s: 0123456789
 Rezultat: 2
```

Primer 4

```
Interakcija sa programom:
 Unesite nisku t: 12345
 Unesite nisku s: 9876543210
 Rezultat: 5
```

Zadatak 2.5.23 Napisati funkciju `int strcspn_klon(char t[], char s[])` koja izračunava dužinu prefiksa niske t sastavljenog isključivo od karaktera koji se ne nalaze u niski s. Napisati program koji testira ovu funkciju za dve unete niske maksimalne dužine 100 karaktera.

Slično važi i za ostale *klon* funkcije iz narednih zadataka.

Primer 1

```
Interakcija sa programom:
 Unesite nisku t:
 programiranje
 Unesite nisku s:
 pero
 Rezultat: 0
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku t:
 programiranje
 Unesite nisku s:
 analiza
 Rezultat: 5
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku t:
 programiranje
 Unesite nisku s:
 1.10.
 Rezultat: 13
```

Zadatak 2.5.24 Napisati funkciju `char* strstr_klon(char s[], char t[])` koja vraća pokazivač na prvo pojavljivanje niske t u niski s ili $NULL$ ukoliko se niska t ne pojavljuje u niski s. Napisati program koji testira napisanu funkciju tako što učitava pet linija i ispisuje redne brojeve svih linija koje sadrže nisku *program*. Ukoliko ne postoji linija sa niskom *program*, ispisati odgovarajuću poruku. Pretpostaviti da je svaka linija maksimalne dužine 100 karaktera kao i da se linije numerišu od broja 1.

Primer 1

```
Interakcija sa programom:
 Unesite pet linija:
 tv program
 c prog. jezik
 c++ programskih jezik
 Programski odbor
 <b>program</b>
 Rezultat: 1 3 5
```

Primer 2

```
Interakcija sa programom:
 Unesite pet linija:
 Programske paradigme
 su predmet na
 trecoj godini
 programerskih
 smerova.
 Rezultat: 4
```

Primer 3

```
Interakcija sa programom:
 Unesite pet linija:
 U narednim
 linijama
 necemo navoditi
 nisku koja se
 trazi.
 Nijedna linija ne sadrzi
 nisku program.
```

Zadatak 2.5.25 Napisati funkciju `int strcmp_klon(char s[], char t[])` koja vraća 0 ako su niske s i t jednake, neku pozitivnu vrednost ako je s leksikografski iza t, a neku negativnu vrednost inače. Napisati program koji učitava dve niske maksimalne dužine 20 karaktera i ako su različite, ispisuje učitane niske u rastućem leksikografskom poretku, a ako su jednake, ispisuje samo jednu nisku.

Primer 1

```
Interakcija sa programom:
 Unesite nisku s: Beograd
 Unesite nisku t: Amsterdam
 Rezultat:
 Amsterdam
 Beograd
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku s: Beograd
 Unesite nisku t: Beograd
 Rezultat:
 Beograd
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku s: radnik
 Unesite nisku t: radnica
 Rezultat:
 radnica
 radnik
```

Zadatak 2.5.26 Napisati funkciju `void obrni(char s[])` koja obrće nisku s. Napisati program koji obrće učitanu nisku maksimalne dužine 20 karaktera i

ispisuje obrnutu nisku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: kisobran
 Rezultat: narbosik
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: Aleksandar
 Rezultat: radnaskelA
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: kajak
 Rezultat: kajak
```

Zadatak 2.5.27 Napisati funkciju `void rotiraj(char s[], int k)` koja rotira nisku s za k mesta ulevo. Napisati program koji učitava nisku maksimalne dužine 20 karaktera i nenegativan ceo broj k i ispisuje rotiranu nisku. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku i broj k:
 sveska 2
 Rezultat: eskasv
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku i broj k:
 olovka 6
 Rezultat: olovka
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku i broj k:
 rezac 8
 Rezultat: acrez
```

Zadatak 2.5.28 Napisati program koji šifruje unetu nisku tako što svako slovo zamenjuje sledećim slovom engelske abecede (slova 'z' i 'Z' zamenjuje, redom, sa 'a' i 'A'), a ostale karaktere ostavlja nepromenjene. Ispisati nisku dobijenu na ovaj način. Pretpostaviti da uneta niska nije duža od 20 karaktera.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: bundeva
 Rezultat: cvoefwb
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: zimzelen
 Rezultat: ajnafmfo
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: Oktobar17
 Rezultat: Plupcbs17
```

Zadatak 2.5.29 Napisati funkciju `void sifruj(char rec[], char sifra[])` koja na osnovu date reči formira šifru tako što se svako slovo u reči zameni sa naredna tri slova engleske abecede (nakon slova 'z' tj. 'Z' sledi slovo 'a' tj. 'A'). Napisati program koji testira napisanu funkciju za reč maksimalne dužine 20 karaktera.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: tamo
 Rezultat: uvwbcdnoppqr
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: Zec
 Rezultat: ABCfghdef
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite nisku: a+b=c
 Rezultat: bcd+cde=def
```

Zadatak 2.5.30 Napisati funkciju `void formiraj(char s1[], char s2[], char c1, char c2)` koja na osnovu niske s_1 formira nisku s_2 udvajanjem svih karaktera c_1 u niski s_1 i izbacivanjem svih karaktera c_2 iz niske s_1, dok ostali karakteri ostaju nepromenjeni. Napisati program koji testira ovu funkciju za unetu nisku i dva uneta karaktera. Pretpostaviti da uneta niska nije duža od 20

karaktera.

Primer 1

```
Interakcija sa programom:
 Unesite nisku: flomaster
 Unesite prvi karakter: s
 Unesite drugi karakter: m
 Rezultat: floasster
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku: bojica
 Unesite prvi karakter: b
 Unesite drugi karakter: a
 Rezultat: bbojic
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku: patentara
 Unesite prvi karakter: t
 Unesite drugi karakter: a
 Rezultat: pttenttr
```

*** Zadatak 2.5.31** Napisati program za rad sa brojevima zapisanim u različitim brojevnim sistemima.

(a) Napisati funkciju `unsigned int u_dekadni_sistem(char broj[], unsigned int osnova)` koja određuje dekadnu vrednost zapisa datog neoznačenog broja *broj* u datoj osnovi *osnova*.

(b) Napisati funkciju `void iz_dekadnog_sistema(unsigned int broj, unsigned int osnova, char rezultat[])` koja datu dekadnu vrednost *broj* zapisuje u datoj osnovi *osnova* i smešta rezultat u nisku *rezultat*. Pretpostaviti da je $0 < osnova \leq 16$.

Napisati program koji učitava broj n koji se zadaje kao niska cifara i osnove o_1 i o_2 i ispisuje dekadnu vrednost broja n u osnovi o_1, kao i zapis tako dobijene dekadne vrednosti u osnovi o_2. Pretpostaviti da je maksimalna dužina zapisa broj 20 karaktera i da će svi brojevi biti ispravno zadati tj. u opsegu tipa `unsigned`.

Primer 1

```
Interakcija sa programom:
 Unesite n, o1 i o2: 10101011 2 16
 Dekadna vrednost broja 10101011: 171
 Vrednost broja 171 u osnovi 16: AB
```

Primer 2

```
Interakcija sa programom:
 Unesite n, o1 i o2: 1067 8 3
 Dekadna vrednost broja 1067: 567
 Zapis broja 567 u osnovi 3: 210000
```

Primer 3

```
Interakcija sa programom:
 Unesite n, o1 i o2: 1010111001010 2 3
 Dekadna vrednost broja 1010111001010: 5578
 Zapis broja 5578 u osnovi 3: 21122121
```

Primer 4

```
Interakcija sa programom:
 Unesite n, o1 i o2: 111 3 5
 Dekadna vrednost broja 111: 13
 Zapis broja 13 u osnovi 5: 23
```

2.6 Rešenja

Rešenje 2.5.1

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/* Poslednji karakter svake niske je terminirajuca nula '\0',
   specijalni karakter ciji je ASCII kod 0.

   Ukoliko je pretpostavka da niska sadrzi najvise 10 karaktera,
   neophodno je deklarisati niz od 11 karaktera, pri cemu se
   dodatni karakter izdvaja za terminirajucu nulu. */
#define MAKS_NISKA 11

/* Funkcija vrsi konverziju svakog malog slova niske u odgovarajuce
   veliko slovo i obrnuto. Ostali karakteri ostaju nepromenjeni. */
void konvertuj(char s[]) {
  int i;

  /* Prolazi se kroz nisku, karakter po karakter, sve dok se ne
     dodje do terminirajuce nule koja sluzi kao oznaka kraja niske.
    */
  for (i = 0; s[i] != '\0'; i++) {
    /* Svako malo slovo se pretvara u veliko i obrnuto. */
    if (islower(s[i]))
      s[i] = toupper(s[i]);
    else if (isupper(s[i]))
      s[i] = tolower(s[i]);
  }

  /* II nacin: Uslov u petlji moze krace da se zapise sa s[i] jer
     ASCII kod terminirajuce nule ima vrednost 0.
     for (i = 0; s[i]; i++) {
       if (islower(s[i]))
         s[i] = toupper(s[i]);
       else if(isupper(s[i]))
         s[i] = tolower(s[i]);
     } */
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_NISKA];

  /* Za razliku od nizova koji se ucitavaju i stampaju element po
     element, niske se mogu ucitati i odstampati pomocu jedne
     scanf/printf naredbe koriscenjem specifikatora %s. */
  printf("Unesite nisku: ");
  scanf("%s", s);
```

```
47
  /* Izmena niske. */
49  konvertuj(s);

51  /* Ispis rezultata. */
  printf("Konvertovana niska: %s\n", s);
53
  exit(EXIT_SUCCESS);
55 }
```

Rešenje 2.5.2

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  #define MAKS_NISKA 21
5
  /* Funkcija ubacuje zvezdice na svako drugo mesto niske s. */
7 void ubaci_zvezdice(char s[]) {
    int i;
9
    for (i = 0; s[i] != '\0' && s[i + 1] != '\0'; i += 2)
11    s[i + 1] = '*';
  }
13
  int main() {
15  /* Deklaracija potrebne promenljive. */
    char s[MAKS_NISKA];
17
    /* Ucitavanje niske. */
19  printf("Unesite nisku: ");
    scanf("%s", s);
21
    /* Izmena niske. */
23  ubaci_zvezdice(s);

25  /* Ispis rezultata. */
    printf("Izmenjena niska: %s\n", s);
27
    exit(EXIT_SUCCESS);
29 }
```

Rešenje 2.5.3

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <ctype.h>

5 #define MAKS_NISKA 21
```

```
/* Funkcija pretvara sva slova niske s u velika slova. */
void u_velika_slova(char s[]) {
  int i;
  for (i = 0; s[i]; i++)
    s[i] = toupper(s[i]);
}

/* Funkcija vraca 1 ako su niske s1 i s2 jednake, a nulu inace. */
int jednake(char s1[], char s2[]) {
  int i;

  /* Prolazi se kroz obe niske dok god ima neobradjenih karaktera u
     bilo kojoj od njih. Ukoliko se naidje na karaktere koji su
     razliciti, kao povratna vrednost se vraca 0 jer u tom slucaju
     niske nisu jednake. */
  for (i = 0; s1[i] || s2[i]; i++)
    if (s1[i] != s2[i])
      return 0;

  /* Ako se doslo do kraja petlje znaci da su se svi karakteri
     poklopili, a da se pri tom doslo do kraja obe niske, tako da
     se kao povratna vrednost funkcije vraca 1 jer su niske s1 i s2
     jednake. */
  return 1;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s1[MAKS_NISKA], s2[MAKS_NISKA];

  /* Ucitavanje niski s1 i s2. */
  printf("Unesite niske:\n");
  scanf("%s%s", s1, s2);

  /* Kako bi se pri poredjenju zanemarila razlika izmedju malih i
     velikih slova, sva slova obe niske se pretvaraju u velika. */
  u_velika_slova(s1);
  u_velika_slova(s2);

  /* Ispis rezultata. */
  if (jednake(s1, s2))
    printf("Niske su jednake.\n");
  else
    printf("Niske nisu jednake.\n");

  exit(EXIT_FAILURE);
}
```

Rešenje 2.5.4

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

#define MAKS_NISKA 21

/* Funkcija proverava da li je karakter c samoglasnik. */
int samoglasnik(char c) {
  /* Karakter se pretvara u veliko slovo kako bi se izbegle posebne
     provere za mala i velika slova. */
  c = toupper(c);

  /* Samoglasnici su slova a, e, i, o i u */
  if (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U')
    return 1;

  return 0;
}

/* Funkcija proverava da li se niska s zavrsava samoglasnikom. */
int samoglasnik_na_kraju(char s[]) {
  /* Funkcija strlen racuna duzinu date niske. Njena deklaracija se
     nalazi u zaglavlju string.h. */
  int duzina = strlen(s);

  /* Ako je niska prazna, ne zavrsava se samoglasnikom. */
  if (duzina == 0)
    return 0;

  /* Provera da li je poslednji karakter niske samoglasnik. */
  return samoglasnik(s[duzina - 1]);
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_NISKA];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Ispis rezultata. */
  if (samoglasnik_na_kraju(s))
    printf("Niska se zavrsava samoglasnikom.\n");
  else
    printf("Niska se ne zavrsava samoglasnikom.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.5 Pogledajte zadatak 2.5.1.

Rešenje 2.5.6

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_NISKA 21

/* Funkcija vraca indeks prvog pojavljivanja karaktera c u okviru
   niske s. Ukoliko se ne pojavljuje, funkcija vraca -1. */
int pozicija(char s[], char c) {
  int i;

  for (i = 0; s[i]; i++)
    if (s[i] == c)
      return i;

  return -1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char s[MAKS_NISKA];
  char c;

  /* Ucitavanje niske i karaktera. */
  printf("Unesite nisku: ");
  scanf("%s", s);
  getchar();
  printf("Unesite karakter: ");
  c = getchar();

  /* I nacin: */
  printf("Pozicija: %d\n", pozicija(s, c));

  /* II nacin: Funkcija strchr(s,c) je funkcija koja vraca adresu
     prvog pojavljivanja karaktera c u niski s, ako se c pojavljuje
     u s, a NULL inace.

     Vrednost promenljive s je zapravo vrednost adrese prvog
     karaktera niske s.

     Ako treba da se ispise indeks prvog pojavljivanja, to moze da
     se uradi tako sto se od adrese koji je vratila funkcija
     strchr oduzme adresa prvog karaktera.

     Na primer:
     s = "koliba" ==> s je adresa karaktera 'k'
     p = strchr(s, 'l') ==> p je adresa karaktera 'l'
     |k|o|l|i|b|a|
      ^   ^
```

```
      |   |
50    s   p
      Izraz p-s ima vrednost 2 (jer je rastojanje izmedju ove dve
52    adrese 2).
      Tip promenljive p je char* jer predstavlja adresu jednog
54    karaktera.

56    char *p = strchr(s, c);
      if (p != NULL)
58      printf("Pozicija: %d\n", p - s);
      else
60      printf("-1\n"); */

62  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.7

```
#include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS_NISKA 11

6 /* Funkcija proverava da li je niska t podniska niske s. */
int podniska(char s[], char t[]) {
8   int i, j;

10  /* Spoljasnja petlja ide redom po niski s. */
    for (i = 0; s[i] != '\0'; i++) {
12    /* Unutrasnja petlja ide redom po niski t pomocu brojaca j i
         proverava da li se cela niska t poklapa sa delom niske s
14       koji pocinje na poziciji i.

16       Cim se naidje na situaciju da se karakteri ne poklapaju,
         izlazi se iz unutrasnje petlje. */
18    for (j = 0; t[j] != '\0'; j++)
        if (s[i + j] != t[j])
20        break;

22    /* Ako je unutrasnja petlja dosla do kraja niske t, to znaci
         da su se svi karakteri iz t poklopili sa karakterima iz s i
24       t je podniska od s. */
      if (t[j] == '\0')
26      return 1;
    }
28  return 0;
}
30

32 int main() {
    /* Deklaracija potrebnih promenljivih. */
```

```
34   char s[MAKS_NISKA], t[MAKS_NISKA];

36   /* Ucitavanje niski s i t. */
     printf("Unesite nisku s: ");
38   scanf("%s", s);
     printf("Unesite nisku t: ");
40   scanf("%s", t);

42   /* Ispis rezultata. */
     if (podniska(s, t))
44     printf("t je podniska niske s.\n");
     else
46     printf("t nije podniska niske s.\n");

48   /* II nacin: Funkcija strstr(t, s) proverava da li je t podniska
        od s i kao povratnu vrednost vraca adresu prvog pojavljivanja
50      t u s ili NULL ukoliko se t ne pojavljuje u s. Deklaracija
        ove funkcije se nalazi u zaglavlju string.h
52
        char* p = strstr(t, s);
54      if(p == NULL)
          printf("t nije podniska od s.\n");
56      else
          printf("t je podniska od s.\n"); */
58
     exit(EXIT_SUCCESS);
60 }
```

Rešenje 2.5.8

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>
   #include <ctype.h>
5
   #define MAKS_LINIJA 101
7
   /* Funkcija ucitava liniju maksimalne duzine n i upisuje je u s.
9     Funkcija ne smesta znak za novi red na kraj linije. */
   void ucitaj_liniju(char s[], int n) {
11   int i = 0, c;

13   /* Ucitavanje karakter po karakter dok se ne unese novi red ili
        oznaka za kraj ulaza ili dok se ne dostigne maksimalan broj
15      karaktera. */
     while ((c = getchar()) != '\n' && i < n - 1 && c != EOF) {
17     s[i] = c;
       i++;
19   }

21   /* Maksimalan broj karaktera za liniju je n-1 jer na kraju treba
```

```
     ostaviti i jedno mesto za terminirajucu nulu. */
  s[i] = '\0';
}

/* Funkcija uklanja beline sa kraja niske s. */
void skrati(char s[]) {
  int i;
  /* Vrsi se prolazak kroz nisku sa desna na levo i trazi se
     pozicija prvog karaktera koji nije belina.

     Funkcija isspace proverava da li je dati karakter neka od
     belina (blanko, tabulator ili novi red) i njena deklaracija se
     nalazi u zaglavlju ctype.h. */
  for (i = strlen(s) - 1; i >= 0; i--)
    if (!isspace(s[i]))
      break;

  /* Nakon izlaska iz petlje, brojac i se nalazi na poziciji prvog
     karaktera sa desne strane koji nije belina. Iz tog razloga se
     na poziciju i+1 upisuje terminirajuca nula kao oznaka da se sada
     tu nalazi kraj niske. */
  s[i + 1] = '\0';
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_LINIJA];

  /* Ucitavanje cele linije sa ulaza. */
  printf("Unesite nisku:\n");
  ucitaj_liniju(s, MAKS_LINIJA);

  /* Napomena: Postoji vise nacina za ucitavanje linije sa
     standardnog ulaza koriscenjem funkcija iz standardne C
     biblioteke. Jedan od njih je koriscenjem funkcije gets:
     gets(s); Postoje razlozi zasto ova funkcija nije bezbedna za
     koriscenje i oni ce biti objasnjeni u kasnijim poglavljima. U
     poglavlju "Datoteke" ce biti predstavljeni i bezbedni nacini
     da se to uradi koriscenjem nekih drugih funkcija. */

  /* Ispis rezultata. */
  printf("Ucitana niska:\n*%s*\n", s);
  skrati(s);
  printf("Izmenjena niska:\n*%s*\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.9

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAKS_LINIJA 21

/* Funkcija uklanja sva slova iz niske s. */
void ukloni_slova(char s[]) {
  int i, j;

  /* Prolazi se kroz nisku s karakter po karakter i vrsi se provera
     da li trenutni karakter treba da se zadrzi. Karakter treba da
     se zadrzi ukoliko nije ni malo ni veliko slovo.

     Brojac j sluzi da pamti gde se upisuje sledeci karakter koji
     treba da se zadrzi i svaki put kada se naidje na takav karaker,
     on se upisuje na poziciju j, a brojac j se uvecava. */
  for (i = 0, j = 0; s[i]; i++)
    if (!islower(s[i]) && !isupper(s[i])) {
      s[j] = s[i];
      j++;
    }

  /* Na kraju se na poziciji j upisuje i terminirajuca nula, kako bi
     se naznacilo da se kraj niske nalazi nakon poslednjeg
     zadrzanog karaktera. */
  s[j] = '\0';
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_LINIJA];

  /* Ucitavanje niske s. */
  printf("Unesite nisku:\n");
  scanf("%s", s);

  /* Ispis rezultata. */
  ukloni_slova(s);
  printf("Rezultat: %s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.10

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <ctype.h>

#define MAKS_LINIJA 101

/* Funkcija ucitava liniju maksimalne duzine n i upisuje je u s.
   Funkcija ne smesta znak za novi red na kraj linije. */
void ucitaj_liniju(char s[], int n) {
  int i = 0, c;

  while ((c = getchar()) != '\n' && i < n - 1 && c != EOF) {
    s[i] = c;
    i++;
  }
  s[i] = '\0';
}

/* Pomocna funkcija koja proverava da li karakter c1 treba zadrzati
   ako vazi da se iza njega nalazi karakter c2. */
int treba_zadrzati(char c1, char c2) {
  /* Ako neki od karaktera nije slovo, c1 se ne izbacuje. */
  if (!isalpha(c1) || !isalpha(c2))
    return 1;

  /* Oba karaktera se pretvaraju u veliko slovo kako bi se smanjio
     broj poredjenja. */
  c1 = toupper(c1);
  c2 = toupper(c2);

  /* c1 se zadrzava ako se c2 ne nalazi iza njega u engleskoj abecedi
     . */
  return c2 <= c1;
}

/* Funkcija uklanja sva slova za koja vazi da se neposredno nakon
   njih nalazi slovo koje je u engleskoj abecedi iza njih. */
void ukloni(char s[]) {
  int i, j;
  for (i = 0, j = 0; s[i]; i++) {
    if (treba_zadrzati(s[i], s[i + 1])) {
      s[j] = s[i];
      j++;
    }
  }
  s[j] = '\0';
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_LINIJA];

  /* Ucitavanje linije sa ulaza. */
  printf("Unesite nisku:\n");
```

```
  ucitaj_liniju(s, MAKS_LINIJA);

  /* Ispis rezultata. */
  ukloni(s);
  printf("Izmenjena niska:\n%s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.11

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 31
#define MAKS_REZULTAT 91

/* Niske se ne kopiraju naredbom dodele. Ukoliko je potrebno da
   neka niska ima isti sadrzaj kao i neka druga niska, moze se
   koristiti funkcija strcpy(t, s) koja kopira karaktere niske s u
   nisku t zajedno za terminirajucom nulom. Deklaracija ove
   funkcije se nalazi u zaglavlju string.h.

   Funkcija strcpy_klon predstavlja jednu implementaciju funkcije
   strcpy. */
void strcpy_klon(char kopija[], char original[]) {
  int i;
  for (i = 0; original[i]; i++)
    kopija[i] = original[i];

  kopija[i] = '\0';
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], t[MAKS_REZULTAT];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Niska s se kopira u nisku t. */
  strcpy_klon(t, s);

  /* Funkcija strcat(s,t) nadovezuje karaktere niske s na kraj
     niske t i novu nisku terminira karakterom '\0'. Deklaracija
     ove funkcije se nalazi u zaglavlju string.h. */

  /* Niska s se jos dva puta nadovezuje na nisku t. */
  strcat(t, s);
```

```
  strcat(t, s);

  /* Ispis rezultata. */
  printf("Rezultujuca niska: %s\n", t);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.12

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 21
#define MAKS_N 10
/* Rezultat se dobija nadovezivanjem niske maksimalne duzine
   MAKS_NISKA-1 i karaktera '-' najvise MAKS_N puta. Odavde je
   maksimalna duzina rezultata: (MAKS_NISKA - 1 + 1) * MAKS_N =
   MAKS_NISKA*MAKS_N. Na ovo treba dodati jos 1 karakter zbog
   terminirajuce nule. */
#define MAKS_REZULTAT (MAKS_NISKA*MAKS_N + 1)

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], t[MAKS_REZULTAT];
  int i, n;

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Ucitavanje broja ponavljanja i provera ispravnosti ulaza. */
  printf("Unesite broj n: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS_N) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Formiranje rezultata. Prvi karakter rezultujuce niske se
     postavlja na terminirajucu nulu. Ovo se radi jer strcat
     funkcionise tako sto krene od pocetka niske, ide do
     terminirajuce nule i zatim pocevsi od tog mesta nadovezuje
     nisku koja je prosledjena kao drugi argument. Na ovaj nacin
     je obezbedjeno da ce prvi poziv funkcije strcat krenuti da
     nadovezuje od pocetka niske t. U petlji se na t nadovezuje prvo
     niska s, a zatim niska "-". Ovo se ponavlja n-1 puta jer nakon
     poslednjeg nadovezivanja niske s ne treba da se nadje "-". Iz
     tog razloga se po zavrsetku petlje vrsi jos jedno nadovezivanje
     niske s, ali ne i niske "-". */
```

```
  t[0] = '\0';
  for (i = 0; i < n - 1; i++) {
    strcat(t, s);
    strcat(t, "-");
  }
  strcat(t, s);

  /* Ispis rezultata. */
  printf("Rezultujuca niska: %s\n", t);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.13

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 21

/* Funkcija kopira prvih n karaktera niske s u nisku t. */
void kopiraj_n(char t[], char s[], int n) {
  int i;
  /* Kopiranje se vrsi ili dok se ne dodje do terminirajuce nule u s
     ili dok se ne prekopira n karaktera. */
  for (i = 0; i < n && s[i] != '\0'; i++)
    t[i] = s[i];

  /* Na kraju rezultujuce niske se upisuje terminirajuca nula. */
  t[i] = '\0';
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n;
  char s[MAKS_NISKA], t[MAKS_NISKA];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Ucitavanje broja n i provera ispravnosti ulaza. */
  printf("Unesite broj n: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS_NISKA - 1) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Formiranje rezultata. */
```

```
  kopiraj_n(t, s, n);

  /* II nacin: Koriscenjem funkcije strncpy(t, s, n), cija se
     deklaracija nalazi u zaglavlju string.h, kopira najvise n
     karaktera niske s u nisku t.

     strncpy(t,s,n); */

  /* Ispis rezultata. */
  printf("Rezultujuca niska: %s\n", t);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.14

```
#include <stdio.h>
#include <stdlib.h>

/* Duzina niske koja se ucitava, bez terminirajuce nule. */
#define MAKS_DUZINA 20

/* Duzine originalne i rezultujuce niske. */
#define MAKS_NISKA (MAKS_DUZINA + 1)
#define MAKS_REZULTAT (2 * MAKS_DUZINA + 1)

/* Funkcija formira nisku t od niske s dupliranjem svakog
   karaktera. Npr. abc postaje aabbcc. */
void dupliranje(char t[], char s[]) {
  int i, j;

  /* Brojac i oznacava tekucu poziciju u niski s, a brojac j
     oznacava tekucu poziciju u niski t. */
  for (i = 0, j = 0; s[i] != '\0'; i++, j += 2) {
    t[j] = s[i];
    t[j + 1] = s[i];

    /* Kraci nacin: t[j] = t[j + 1] = s[i]; */
  }

  /* Upisuje se terminirajuca nula na kraj rezultujuce niske. */
  t[j] = '\0';
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], t[MAKS_REZULTAT];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);
```

```
37   /* Formiranje niske t. */
     dupliranje(t, s);
39
     /* Ispis rezultata. */
41   printf("Rezultujuca niska: %s\n", t);

43   exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.15

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <ctype.h>

5  #define MAKS_NISKA 10

7  /* Funkcija formiraj_broj na osnovu niske b formira ceo broj ciji
      je to zapis.
9
      Ako su cifre broja a, b, c i d, tada broj mozemo formirati kao:
11    a*10^3 + b*10^2 + c*10^1 + d*10^0. Medjutim, efikasnije je
      koristiti Hornerovu semu: 10*(10*(10*(10*0 + a)+b)+c)+d. */
13 int formiraj_broj(char b[]) {
     int i;
15   int broj = 0, znak;

17   /* Odredjivanje znaka broja i pozicije prve cifre. */
     if (b[0] == '-') {
19     znak = -1;
       i = 1;
21   } else if (b[0] == '+') {
       znak = 1;
23     i = 1;
     } else {
25     i = 0;
       znak = 1;
27   }

29   /* Prolazak kroz cifre broja i racunanje vrednosti broja
        koriscenjem Hornerove sheme. Vrednost trenutne cifre se dobija
31      kada se od trenutnog karaktera (b[i]) oduzme karakter '0'.
        Ako se naidje na karakter koji nije cifra, petlja se prekida.
33      Na primer, za b="123abc", rezultat treba da bude 123. */
     for (; b[i] != '\0'; i++) {
35     if (isdigit(b[i]))
         broj = broj * 10 + (b[i] - '0');
37     else
         break;
39   }
```

```
41    return broj * znak;
  }
43
  int main() {
45    /* Deklaracija potrebne promenljive. */
      char s[MAKS_NISKA];
47
      /* Broj se ucitava kao niska. */
49    printf("Unesite nisku: ");
      scanf("%s", s);
51
      /* Ispis rezultata. */
53    printf("Rezultat: %d\n", formiraj_broj(s));

55    /* II nacin: Koriscenjem funkcije atoi. Deklaracija ove funkcije
         se nalazi u zaglavlju stdlib.h.
57
         printf("%d\n", atoi(s)); */
59
      exit(EXIT_SUCCESS);
61  }
```

Rešenje 2.5.16

```
1   #include <stdio.h>
    #include <stdlib.h>
3
    #define MAKS_NISKA 10
5
    /* Funkcija racuna broj cifara broja n. */
7   int broj_cifara(int n) {
      int brojac = 0;
9     do {
        brojac++;
11      n /= 10;
      } while (n);
13
      return brojac;
15  }

17  /* Funkcija od prosledjenog broja formira nisku. */
    void broj_u_nisku(int broj, char s[]) {
19    int n, cifra, i;

21    /* Promenljiva n cuva informaciju o duzini niske. Duzina niske
         odgovara broju cifara prosledjenog broja. Ukoliko je broj
23       negativan, onda se duzina uvecava za 1 i na prvo mesto se
         upisuje znak '-'. */
25    n = broj_cifara(broj);
      if (broj < 0) {
```

```
    s[0] = '-';
    n++;
  }

  /* U nastavku se radi sa apsolutnom vrednoscu broja. */
  broj = abs(broj);

  /* Cifre broja se upisuju u nisku s sa desna na levo. */
  s[n] = '\0';
  i = n - 1;
  do {
    /* Karakter koji odgovara trenutnoj cifri se dobija izrazom '0'
       + cifra. Na primer, '0' + 5 je '5' jer se karakter '5'
       nalazi 5 mesta nakon karaktera '0' u ASCII tablici. */
    cifra = broj % 10;
    broj = broj / 10;
    s[i] = '0' + cifra;
    i--;
  } while (broj);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n;
  char s[MAKS_NISKA];

  /* Ucitavanje broja. */
  printf("Unesite ceo broj: ");
  scanf("%d", &n);

  /* Formiranje niske. */
  broj_u_nisku(n, s);

  /* Ispis rezultata. */
  printf("Rezultat: %s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.17

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAKS_NISKA 8

/* Funkcija proverava da li je prosledjeni karakter ispravna
   heksadekadna cifra. */
int heksa_cifra(char c) {
  c = toupper(c);
```

```
11
  /* Karakter je ispravan ako je cifra ili ako je neko od slova:
13    A, B, C, D, E ili F. */
  return isdigit(c) || (c >= 'A' && c <= 'F');
15 }

17 /* Funkcija proverava da li prosledjena niska s predstavlja
   ispravan heksadekadni broj. */
19 int heksadekadni_broj(char s[]) {
  int i;
21
  /* Svaki heksadekasni broj pocinje sa 0x ili 0X. */
23  if (s[0] != '0' || toupper(s[1]) != 'X')
    return 0;
25
  /* Za svaki karakter niske s se proverava da li predstavlja
27    ispravnu heksadekadnu cifru. Ako se naidje na karakter koji
    ne zadovoljava taj uslov, onda se kao povratna vrednost vraca
29    nula. */
  for (i = 2; s[i]; i++)
31    if (!heksa_cifra(s[i]))
      return 0;
33
  /* Ako su svi karakteri isravne heksadekadne cifre, onda je i s
35    ispravan heksadekadni broj i funkcija vraca jedninicu. */
  return 1;
37 }

39 int main() {
  /* Deklaracija potrebne promenljive. */
41  char s[MAKS_NISKA];

43  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
45  scanf("%s", s);

47  /* Ispis rezultata. */
  if (heksadekadni_broj(s))
49    printf("Korektan heksadekadni broj.\n");
  else
51    printf("Nekorektan heksadekadni broj.\n");

53  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.18

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
  #include <ctype.h>
```

```
5
  #define MAKS_NISKA 8
7
  /* Funkcija racuna dekadnu vrednost jedne heksadekadne cifre. Ako
9    je c dekadna cifra, vrednost se dobija oduzimanjem '0'. Ako je c
     veliko slovo, vrednost se dobija oduzimanjem 'A' i dodavanjem 10
11   (npr. vrednost karaktera 'B' je 10 + 'B' - 'A' = 11). */
  int vrednost_heksa_cifre(char c) {
13   if (isdigit(c))
       return c - '0';
15   else
       return 10 + toupper(c) - 'A';
17 }

19 /* Funkcija racuna dekadnu vrednost heksadekadnog broja. */
  int dekadna_vrednost(char s[]) {
21   int i, tezina_pozicije = 1, rezultat = 0;
     int n = strlen(s);
23
     /* Vrsi se prolazak kroz nisku sa desna na levo. Heksadekadna
25      cifra najvece tezine se nalazi na poziciji n-1, a cifra najmanje
        tezine se nalazi na poziciji 2 (jer su prva dva karaktera 0x).
27
        U svakoj iteraciji, na rezultat se dodaje vrednost tekuce
29      cifre pomnozene vrednoscu tezine njene pozicije.
        Na primer, za s = "0x1a8e", n=6
31      i = 5, rezultat += vrednost('e')*1 => rezultat += 11*1
        i = 4, rezultat += vrednost('8')*16 => rezultat += 8*16
33      i = 3, rezultat += vrednost('a')*256 => rezultat += 10*256
        i = 2, rezultat += vrednost('1')*4096 => rezultat += 1*4096 */
35   for (i = n - 1; i >= 2; i--) {
       rezultat += tezina_pozicije * vrednost_heksa_cifre(s[i]);
37     tezina_pozicije *= 16;
     }
39
     /* II nacin: Koriscenjem Hornerove sheme.
41   for (i = 2; i < n; i++)
       rezultat = rezultat * 16 + vrednost_heksa_cifre(s[i]); */
43
     return rezultat;
45 }

47 int main() {
     /* Deklaracija potrebne promenljive. */
49   char s[MAKS_NISKA];

51   /* Ucitavanje niske. */
     printf("Unesite nisku: ");
53   scanf("%s", s);

55   /* Ispis rezultata. */
     printf("Rezultat: %d\n", dekadna_vrednost(s));
```

```
57
  exit(EXIT_SUCCESS);
59 }
```

Rešenje 2.5.19

```
#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAKS_LINIJA 81
6
/* Funkcija ucitava liniju maksimalne duzine n i upisuje je u s.
8    Funkcija ne smesta znak za novi red na kraj linije. */
int ucitaj_liniju(char s[], int n) {
10   int i = 0;
  int c;
12
  /* Ucitava se karakter po karakter dok se ne unese novi red ili
14     oznaka za kraj ulaza ili dok se ne dostigne maksimalan broj
     karaktera. */
16   while ((c = getchar()) != '\n' && i < n - 1 && c != EOF) {
    s[i] = c;
18     i++;
  }
20
  /* Maksimalan broj karaktera za liniju je n-1 jer na kraju treba
22     ostaviti i jedno mesto za terminirajucu nulu. */
  s[i] = '\0';
24
  return i;
26 }

28 int main() {
  /* Deklaracije potrebnih promenljivih. */
30   char linija[MAKS_LINIJA], najduza_linija[MAKS_LINIJA];
  int duzina_najduze = 0, duzina;
32
  /* U petlji se ucitavaju linije sve dok se ne unese prazna
34     linija. Ukoliko se unese linija koja je duza od trenutno
     najduze, vrsi se azuriranje duzine najduze linije, kao i same
36     linije. */
  printf("Unesite tekst:\n");
38   while ((duzina = ucitaj_liniju(linija, MAKS_LINIJA)) > 0)
    if (duzina_najduze < duzina) {
40       duzina_najduze = duzina;
       strcpy(najduza_linija, linija);
42     }

44   /* Ispis rezultata. */
  if (duzina_najduze == 0)
```

```
46      printf("Nije uneta nijedna linija.\n");
    else
48      printf("Najduza linija:\n%s\nDuzina: %d\n", najduza_linija,
              duzina_najduze);
50
    exit(EXIT_SUCCESS);
52  }
```

Rešenje 2.5.20

```
#include <stdio.h>
2  #include <stdlib.h>
#include <string.h>
4  #include <ctype.h>

6  #define MAKS_RECENICA 81

8  /* Funkcija ucitava recenicu maksimalne duzine n. */
int ucitaj_recenicu(char s[], int n) {
10   int i = 0, c;

12   /* Ako postoje, preskacu se beline sa pocetka. Po zavrsetku ove
        petlje u c se nalazi prvi sledeci karakter koji nije belina. */
14   do {
       c = getchar();
16   } while (isspace(c));

18   /* Ako je taj karakter EOF, zavrsava se ucitavanje. */
     if (c == EOF)
20     return 0;

22   /* U nisku se smesta karakter, prelazi se na sledeci karakter i
        postupak se ponavlja sve dok se ne unese tacka, EOF ili dok se
24      ne smesti maksimalan broj karaktera koje recenica moze da
        sadrzi. */
26   do {
       s[i] = c;
28     i++;
       c = getchar();
30   } while (c != '.' && i < n - 2 && c != EOF);

32   /* Ako je poslednji uneti karakter EOF, zavrsava se ucitavanje. */
     if (c == EOF)
34     return 0;

36   /* Na kraju svake recenice stoji tacka za kojom sledi '\0'. */
     s[i] = '.';
38   s[i + 1] = '\0';

40   return i + 1;
}
```

```
42
   /* Funkcija prebrojava mala i velika slova. */
44 void prebroj(char s[], int *broj_malih, int *broj_velikih) {
     int i, mala = 0, velika = 0;
46
     for (i = 0; s[i]; i++) {
48     if (islower(s[i]))
         mala++;
50     else if (isupper(s[i]))
         velika++;
52   }

54   *broj_malih = mala;
     *broj_velikih = velika;
56 }

58 int main() {
     /* Deklaracija potrebnih promenljivih. */
60   char recenica[MAKS_RECENICA];
     char rezultujuca_recenica[MAKS_RECENICA];
62   int najveca_razlika = -1, trenutna_razlika;
     int mala, velika;
64   int ucitana_bar_jedna = 0;

66   /* U petlji se ucitavaju recenice sve dok se ne unese EOF. */
     while (ucitaj_recenicu(recenica, MAKS_RECENICA) > 0) {
68     /* Prebrojavanje malih i velikih slova. */
       prebroj(recenica, &mala, &velika);
70
       /* Racunanje njihove apsolutne razlike. */
72     trenutna_razlika = abs(mala - velika);

74     /* Ako je razlika veca od trenutno najvece, azurira se vrednost
          najvece razlike i pamti se trenutna recenica. */
76     if (trenutna_razlika > najveca_razlika) {
         najveca_razlika = trenutna_razlika;
78       strcpy(rezultujuca_recenica, recenica);
       }
80
       /* Indikator koji oznacava da se petlja bar jednom izvrsila,
82        tj. da korisnik nije odmah zadao EOF. */
       ucitana_bar_jedna = 1;
84   }

86   /* Ispis rezultata. */
     if (ucitana_bar_jedna)
88     printf("Rezultujuca recenica:\n%s\n", rezultujuca_recenica);
     else
90     printf("Nije uneta nijedna recenica. ");

92   exit(EXIT_SUCCESS);
   }
```

Rešenje 2.5.21

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_NISKA 21

/* Funkcija vraca adresu prvog pojavljivanja karaktera c u niski s
   ili NULL ukoliko se c ne pojavljuje u s.

   Trazeni rezultat se moze dobiti koriscenjem funkcije strchr cija
   se deklaracija nalazi u zaglavlju string.h. Funkcija
   strchr_klon predstavlja jednu mogucu implementaciju ove
   funkcije. */
char *strchr_klon(char s[], char c) {
  int i;

  /* Za svaki karakter se proverava da li je jednak karakteru c.
     Ako se naidje na takav karakter, kao povratna vrednost
     funkcije se vraca njegova adresa (&s[i]). */
  for (i = 0; s[i]; i++)
    if (s[i] == c)
      return &s[i];

  /* Ako je petlja zavrsena, znaci da nije pronadjen karakter koji
     je jednak karakteru c pa se kao povratna vrednost funkcije
     vraca NULL pokazivac. */
  return NULL;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char s[MAKS_NISKA];
  char c;

  /* Ucitavanje niske s. */
  printf("Unesite nisku s: ");
  scanf("%s", s);

  /* Preskace se novi red koji je unet nakon niske s i ucitava se
     karakter c. */
  getchar();
  printf("Unesite karakter c: ");
  scanf("%c", &c);

  /* Racunanje i ispis rezultata. */
  char *p = strchr_klon(s, c);
  if (p == NULL)
    printf("Pozicija: -1\n");
  else
    printf("Pozicija: %ld\n", p - s);

```

```
  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.22

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 21

/* Funkcija racuna duzinu prefiksa niske t koji se moze zapisati
   pomocu karaktera niske s. Na primer, t="programiranje",
   s="grupacija", rezultat je 2 jer niska s sadrzi prva dva
   karaktera niske t, ali ne i treci.
   Trazeni rezultat moze se dobiti koriscenjem funkcije strspn cija
   se deklaracija nalazi u zaglavlju string.h. Funkcija
   strspn_klon predstavlja jednu mogucu implementaciju ove
   funkcije. */
int strspn_klon(char t[], char s[]) {
  int i, brojac = 0;

  /* Ide se redom po karakterima niske t i za svaki karakter se
     vrsi provera da li se on nalazi u zapisu niske s. Za ovo se
     koristi funkcija strchr. Ako se nalazi, uvecava se brojac, a
     ako se ne nalazi, prekida se petlja. */
  for (i = 0; t[i]; i++) {
    if (strchr(s, t[i]) != NULL)
      brojac++;
    else
      break;
  }

  return brojac;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], t[MAKS_NISKA];

  /* Ucitavanje niski. */
  printf("Unesite nisku t: ");
  scanf("%s", t);
  printf("Unesite nisku s: ");
  scanf("%s", s);

  /* Racunanje i ispis rezultata. */
  printf("Rezultat: %d\n", strspn_klon(t, s));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.23 Rešenje ovog zadatka se svodi na rešenje zadatka 2.5.22, uz razliku da se ovde prebrojavaju karakteri koji se ne nalaze u zapisu niske **s**.

Rešenje 2.5.24

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 101

/* Funkcija ucitava liniju maksimalne duzine n i upisuje je u s.
   Funkcija ne smesta znak za novi red na kraj linije. */
void ucitaj_liniju(char s[], int n) {
  int i = 0, c;

  while ((c = getchar()) != '\n' && i < n - 1 && c != EOF) {
    s[i] = c;
    i++;
  }

  s[i] = '\0';
}

/* Funkcija vraca pokazivac na prvo pojavljivanje niske t u okviru
   niske s ili NULL ukoliko se t ne nalazi u s.

   Trazeni rezultat moze se dobiti koriscenjem funkcije strstr cija
   se deklaracija nalazi u zaglavlju string.h. Funkcija
   strstr_klon predstavlja jednu mogucu implementaciju ove
   funkcije. */
char *strstr_klon(char s[], char t[]) {
  int i, j;

  /* Spoljasnja petlja ide redom po niski s. */
  for (i = 0; s[i] != '\0'; i++) {
    /* Unutrasnja petlja ide redom po niski t pomocu brojaca j i
       proverava da li se cela niska t poklapa sa delom niske s
       koji pocinje na poziciji i.
       Cim se naidje na situaciju da se karakteri ne poklapaju,
       izlazi se iz unutrasnje petlje. */
    for (j = 0; t[j] != '\0'; j++)
      if (s[i + j] != t[j])
        break;

    /* Ako je unutrasnja petlja dosla do kraja niske t, to znaci
       da su se svi karakteri iz t poklopili sa karakterima iz s i
       t je podniska od s. Kao povratna vrednost se vraca adresa
       gde t pocinje u s. */
    if (t[j] == '\0')
      return &s[i];
  }
```

```
49    return NULL;
  }
51
  int main() {
53    /* Deklaracije potrebnih promenljivih. */
      char linija[MAKS_NISKA];
55    int i, bar_jedna = 0;

57    /* Ucitavanje linija i ispis rednih brojeva linija koje sadrze
         rec "program". */
59    printf("Unesite pet linija:\n");
      for (i = 1; i <= 5; i++) {
61      ucitaj_liniju(linija, MAKS_NISKA);
        if (strstr_klon(linija, "program") != NULL) {
63        if(!bar_jedna)
            printf("Rezultat: ");
65        printf("%d ", i);
          bar_jedna = 1;
67      }
        /* II nacin: Koriscenjem funkcije strstr cija se deklaracija
69         nalazi u zaglavlju string.h.
           if(strstr(linija, "program") != NULL){
71           printf("%d ", i);
             bar_jedna = 1;
73         } */
      }
75    printf("\n");

77    /* Ako indikator bar_jedna i dalje ima vrednost 0, znaci da nije
         uneta nijedna linija koja sadrzi rec "program". */
79    if (!bar_jedna)
        printf("Nijedna linija ne sadrzi nisku program.\n");
81
      exit(EXIT_SUCCESS);
83  }
```

Rešenje 2.5.25

```
  #include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS_NISKA 21

6 /* Funkcija poredi dve niske i vraca nulu ukoliko su jednake, neku
     pozitivnu vrednost ukoliko je niska s1 leksikografski iza s2,
8    a neku negativnu vrednost inace.

10   Trazeni rezultat moze se dobiti koriscenjem funkcije strcmp cija
     se deklaracija nalazi u zaglavlju string.h. Funkcija
12   strcmp_klon predstavlja jednu mogucu implementaciju ove
```

```
   funkcije. */
int strcmp_klon(char s1[], char s2[]) {
  int i;

  /* Prolazi se kroz obe niske dok god se odgovarajuci karakteri
     poklapaju. Ako se u ovom prolasku desi da je petlja dosla do
     kraja obe niske, onda su one jednake i kao povratna vrednost
     funkcije se vraca 0. */
  for (i = 0; s1[i] == s2[i]; i++)
    if (s1[i] == '\0')
      return 0;

  /* Ako niske nisu jednake, znaci da je brojac i stao na prvom
     mestu gde se niske s1 i s2 razlikuju. Posto funkcija treba da
     vrati pozitivnu vrednost ako je niska s1 laksikografski iza
     s2, a negativnu u suprotnom, ovo moze biti realizovano
     vracanjem razlike ASCII kodova. Na primer: s1 = "pero", s2 =
     "program" Nakon petlje, brojac i ima vrednost 1 (jer je tu
     prva razlika). Kao povratna vrednost se vraca s1[1] - s2[1] =
     'e' - 'r' = -13 sto kao negativna vrednost govori da se s1
     nalazi leksikografski ispred s2. */
  return s1[i] - s2[i];
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char s[MAKS_NISKA], t[MAKS_NISKA];
  int rezultat;

  /* Ucitavanje niski s i t. */
  printf("Unesite nisku s: ");
  scanf("%s", s);
  printf("Unesite nisku t: ");
  scanf("%s", t);

  /* Poredjenje niski i ispis rezultata. */
  rezultat = strcmp_klon(s, t);

  /* II nacin: Koriscenjem funkcije strcmp cija se deklaracija
     nalazi u zaglavlju string.g: rezultat = strcmp(s, t); */

  /* Ispis rezultata. */
  printf("Rezultat:\n");
  if (rezultat == 0)
    printf("%s\n", s);
  else if (rezultat < 0)
    printf("%s\n%s\n", s, t);
  else
    printf("%s\n%s\n", t, s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.26

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 21

/* Funkcija obrce nisku s. */
void obrni(char s[]) {
  int i, j;
  int n = strlen(s);
  char c;

  /* Brojac i ide od prvog karaktera niske s, a brojac j od
     poslednjeg i dok god se ne sretnu, vrsi se zamena karaktera
     koji se nalaze na njihovim pozicijama. */
  for (i = 0, j = n - 1; i < j; i++, j--) {
    c = s[i];
    s[i] = s[j];
    s[j] = c;
  }
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_NISKA];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Racunanje i ispis rezultata. */
  obrni(s);
  printf("Rezultat: %s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.27

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_NISKA 21

/* Funkcija rotira nisku za jedno mesto ulevo. */
void rotiraj1(char s[], int n) {
  int i;
  /* Pamti se prvi karakter. */
```

```
  char prvi = s[0];

  /* Svaki sledeci karakter se pomera za jedno mesto ulevo. */
  for (i = 0; i < n - 1; i++)
    s[i] = s[i + 1];

  /* Prvi karakter se upisuje na kraj niske. */
  s[n - 1] = prvi;
}

/* Funkcija rotira nisku s za k mesta ulevo. */
void rotiraj(char s[], int k) {
  int i;
  int n = strlen(s);

  for (i = 0; i < k; i++)
    rotiraj1(s, n);
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA];
  int k;

  /* Ucitavanje niske i vrednosti broja k. */
  printf("Unesite nisku i broj k: ");
  scanf("%s%d", s, &k);

  /* Provera ispravnosti ulaza. */
  if (k < 0) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Racunanje i ispis rezultata. */
  rotiraj(s, k);
  printf("Rezultat: %s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.28

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAKS_NISKA 21

/* Funkcija svako slovo niske s menja slovom koje se u ASCII
```

```
   tablici nalazi neposredno iza njega. Specijalan slucaj je slovo
   z koje treba da se zameni slovom a. Ostali karakteri ostaju
   nepromenjeni. */
void sifruj(char s[]) {
  int i;

  for (i = 0; s[i]; i++)
    if (isalpha(s[i])) {
      if (s[i] == 'z')
        s[i] = 'a';
      else if (s[i] == 'Z')
        s[i] = 'A';
      else
        s[i] = s[i] + 1;
    }
}

int main() {
  /* Deklaracija potrebne promenljive. */
  char s[MAKS_NISKA];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Racunanje i ispis rezultata. */
  sifruj(s);
  printf("Rezultat: %s\n", s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.29

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAKS_DUZINA 20
#define MAKS_NISKA (MAKS_DUZINA + 1)
#define MAKS_REZULTAT (3*MAKS_DUZINA + 1)

/* Pomocna funkcija koja za prosledjeno slovo vraca slovo koje ide
   posle njega. */
char sledeci(char c) {
  if (c == 'z')
    return 'a';

  if (c == 'Z')
    return 'A';

```

```
  return c + 1;
}

/* Funkcija od niske s formira rezultujucu nisku koja se dobija na
   sledeci nacin:
   1. ako je s[i] slovo, onda se u rezultujucu nisku upisuju naredna
   tri slova engelske abecede (kada se stigne do kraja engleske
    abecede, ide se u
   krug, tj. nakon slova z sledi slovo a)
   2. ako s[i] nije slovo, s[i] se samo prepisuje u rezultat. */
void sifruj(char s[], char rezultat[]) {
  int i, j;

  /* Brojac i se koristi za nisku s, a brojac j za rezultujucu
     nisku. */
  for (i = 0, j = 0; s[i]; i++) {
    if (isalpha(s[i])) {
      /* Ako je s[i] slovo, onda se u rezultat upisuju 3 slova koja
         slede nakon njega. */
      rezultat[j] = sledeci(s[i]);
      rezultat[j + 1] = sledeci(rezultat[j]);
      rezultat[j + 2] = sledeci(rezultat[j + 1]);
      j += 3;
    } else {
      /* Ako s[i] nije slovo, onda se samo prepisuje u rezultat. */
      rezultat[j] = s[i];
      j++;
    }
  }

  /* Na kraj rezultata se dopisuje terminirajuca nula. */
  rezultat[j] = '\0';
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], rezultat[MAKS_REZULTAT];

  /* Ucitavanje niske. */
  printf("Unesite nisku: ");
  scanf("%s", s);

  /* Racunanje i ispis rezultata. */
  sifruj(s, rezultat);
  printf("Rezultat: %s\n", rezultat);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.5.30

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAKS_DUZINA 20
#define MAKS_NISKA (MAKS_DUZINA + 1)
#define MAKS_REZULTAT (2*MAKS_DUZINA + 1)

/* Funkcija od niske s formira rezultujucu nisku na sledeci nacin:
   1. Svi karakteri niske s koji su jednaki c1 se dupliraju. 2. Svi
   karakteri niske s koji su jednaki c2 se brisu. 3. Ostali
   karakteri se samo prepisuju. */
void formiraj(char s[], char rezultat[], char c1, char c2) {
  int i, j;
  /* Brojac i se koristi za nisku s, a brojac j za rezultujucu
     nisku. */
  for (i = 0, j = 0; s[i]; i++) {
    if (s[i] == c1) {
      /* Ako je s[i] jednako c1, duplira se u rezultatu. */
      rezultat[j] = s[i];
      rezultat[j + 1] = s[i];
      j += 2;
    } else if (s[i] != c2) {
      /* Ako s[i] razlicito od c2, upisuje se u rezultat. */
      rezultat[j] = s[i];
      j++;
    }
  }

  /* Na kraj rezultata se dopisuje terminirajuca nula. */
  rezultat[j] = '\0';
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAKS_NISKA], rezultat[MAKS_REZULTAT];
  char c1, c2;

  /* Ucitavanje niske i karaktera. */
  printf("Unesite nisku: ");
  scanf("%s", s);
  getchar();
  printf("Unesite prvi karakter: ");
  scanf("%c", &c1);
  getchar();
  printf("Unesite drugi karakter: ");
  scanf("%c", &c2);

  /* Provera ispravnosti ulaza. */
  if (c1 == c2) {
```

```
51      printf("Greska: neispravan unos.\n");
        exit(EXIT_FAILURE);
53    }

55    /* Racunanje i ispis rezultata. */
      formiraj(s, rezultat, c1, c2);
57    printf("Rezultat: %s\n", rezultat);

59    exit(EXIT_SUCCESS);
  }
```

Rešenje 2.5.31

```
  #include <stdio.h>
2 #include <stdlib.h>
  #include <string.h>
4 #include <ctype.h>

6 #define MAKS_NISKA 20

8 /* Pomocna funkcija koja racuna dekadnu vrednost prosledjenog
     karaktera (npr. '1' ima vrednost 1, 'C' ima vrednost 12). */
10 unsigned vrednost_cifre(char c) {
    c = toupper(c);
12  if (isdigit(c))
      return c - '0';
14  else
      return c - 'A' + 10;
16 }

18 /* Funkcija racuna dekadnu vrednost neoznacenog broja zapisanog u
     datoj osnovi. */
20 unsigned int u_dekadni_sistem(char broj[], unsigned int osnova) {
    int i, n = strlen(broj);
22  int rezultat = 0, tezina_pozicije = 1;

24  for (i = n - 1; i >= 0; i--) {
      rezultat += vrednost_cifre(broj[i]) * tezina_pozicije;
26    tezina_pozicije *= osnova;
    }
28
    return rezultat;
30 }

32 /* Funkcija obrce nisku s. */
  void obrni(char s[]) {
34  int i, j;
    int n = strlen(s);
36  char c;

38  for (i = 0, j = n - 1; i < j; i++, j--) {
```

```
    c = s[i];
    s[i] = s[j];
    s[j] = c;
  }
}

/* Pomocna funkcija koja dekadnu vrednost cifre pretvara u
   odgovarajuci karakter (npr. 12 u 'C', 5 u '5'). */
char ostatak_u_char(int ostatak) {
  if (ostatak < 10)
    return '0' + ostatak;
  else
    return 'A' + ostatak - 10;
}

/* Funkcija datu dekadnu vrednost broja prebacuje u broj u datoj
   osnovi. */
void iz_dekadnog_sistema(unsigned int broj, unsigned int osnova,
                         char rezultat[]) {
  int i = 0;
  int ostatak;

  do {
    ostatak = broj % osnova;
    broj = broj / osnova;
    rezultat[i] = ostatak_u_char(ostatak);
    i++;
  } while (broj);

  rezultat[i] = '\0';
  obrni(rezultat);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char broj[MAKS_NISKA], broj2[MAKS_NISKA];
  unsigned int osnova1, osnova2;

  /* Ucitavanje ulaznih podataka. */
  printf("Unesite n, o1 i o2: ");
  scanf("%s%u%u", broj, &osnova1, &osnova2);

  /* Ispis rezultata. */
  unsigned dekadna_vrednost = u_dekadni_sistem(broj, osnova1);
  printf("Dekadna vrednost broja %s: %u\n", broj, dekadna_vrednost);

  iz_dekadnog_sistema(dekadna_vrednost, osnova2, broj2);
  printf("Zapis broja %u u osnovi %u: %s\n", dekadna_vrednost,
         osnova2, broj2);

  exit(EXIT_SUCCESS);
}
```

2.7 Višedimenzioni nizovi

Zadatak 2.7.1 Napisati program koji učitava i zatim ispisuje elemente učitane matrice. Sa ulaza se najpre učitavaju dva cela broja m i n, a potom i elementi matrice celih brojeva dimenzije $m \times n$. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Matrica je:
 1 2 3 4
 5 6 7 8
 9 10 11 12
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 5 3
 Unesite elemente matrice:
 1 1 2
 5 0 2
 7 8 9
 1 2 4
 0 1 1
 Matrica je:
 1 1 2
 5 0 2
 7 8 9
 1 2 4
 0 1 1
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 500 3
 Greska: neispravan unos.
```

Zadatak 2.7.2 Napisati program koji za učitanu celobrojnu matricu[2] dimenzije $m \times n$ izračunava i štampa na tri decimale njenu Ekulidsku normu. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. Uputstvo: *Euklidska norma matrice je kvadratni koren sume kvadrata svih elemenata matrice.*

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Euklidska norma: 25.495
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 5 3
 Unesite elemente matrice:
 1 1 2
 5 0 2
 7 8 9
 1 2 4
 0 1 1
 Euklidska norma: 15.875
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 500 3
 Greska: neispravan unos.
```

Zadatak 2.7.3 Napisati funkcije za rad sa celobrojnim matricama:

[2]Pod pojmom *učitati matricu* ili *za datu matricu* uvek se podrazumeva da se prvo unose dimenzije matrice, a potom i sama matrica.

(a) `void ucitaj(int a[][MAKS], int n, int m)` kojom se učitavaju elementi matrice celih brojeva a dimenzije $m \times n$,

(b) `void ispisi(int a[][MAKS], int n, int m)` kojom se ispisuju elementi matrice a dimenzije $m \times n$.

Napisati program koji najpre učitava, a zatim i ispisuje elemente učitane matrice. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *U ovom i u narednim zadacima, konstanta* `MAKS` *u prototipu funkcije označava maksimalni broj kolona date matrice i potrebno ju je definisati u rešenju direkivom* `#define`.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Matrica je:
 1 2 3 4
 5 6 7 8
 9 10 11 12
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 2 5
 Unesite elemente matrice:
 1 1 2 3 4
 5 0 2 5 7
 Matrica je:
 1 1 2 3 4
 5 0 2 5 7
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 500 3
 Greska: neispravan unos.
```

Zadatak 2.7.4 Napisati funkciju `void transponovana(int a[][MAKS], int m, int n, int b[][MAKS])` koja određuje matricu b koja je dobijena transponovanjem matrice a. Napisati program koji za učitanu matricu celih brojeva ispisuje odgvarajuću transponovanu matricu. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Transponovana matrica je:
 1 5 9
 2 6 10
 3 7 11
 4 8 12
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 5 3
 Unesite elemente matrice:
 1 1 2
 5 0 2
 7 8 9
 1 2 4
 0 1 1
 Transponovana matrica je:
 1 5 7 1 0
 1 0 8 2 1
 2 2 9 4 1
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i
 broj kolona matrice:
 500 3
 Greska: neispravan unos.
```

Zadatak 2.7.5 Napisati funkciju `void razmeni(int a[][MAKS], int m, int n, int k, int t)` u kojoj se razmenjuju elemeti k–te i t–te vrste matrice a dimezije $m \times n$. Napisati program koji za učitanu matricu celih brojeva i dva cela broja k i t ispisuje matricu dobijenu razmenjivanjem k–te i t–te vrste ulazne matrice. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
  Unesite broj vrsta i
  broj kolona matrice:
  3 4
  Unesite elemente matrice:
  1 2 3 4
  5 6 7 8
  9 10 11 12
  Unesite indekse vrsta:
  0 2
  Rezultujuca matrica:
  9 10 11 12
  5 6 7 8
  1 2 3 4
```

Primer 2

```
Interakcija sa programom:
  Unesite broj vrsta i
  broj kolona matrice:
  5 3
  Unesite elemente matrice:
  1 1 2
  5 0 2
  7 8 9
  1 2 4
  0 1 1
  Unesite indekse vrsta:
  1 3
  Rezultujuca matrica:
  1 1 2
  1 2 4
  7 8 9
  5 0 2
  0 1 1
```

Primer 3

```
Interakcija sa programom:
  Unesite broj vrsta i
  broj kolona matrice:
  5 3
  Unesite elemente matrice:
  1 1 2
  5 0 2
  7 8 9
  1 2 4
  0 1 1
  Unesite indekse vrsta:
  -1 50
  Greska: neispravan unos.
```

Zadatak 2.7.6 Napisati program koji za učitanu matricu celih brojeva ispisuje indekse onih elemenata matrice koji su jednaki zbiru svih svojih susednih elemenata. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

```
- - - - - - s b s -
- s s s - - s s s -
- s a s - - - - - -
- s s s - - - - - -
- - - - - - - - s s
- - - - - - - - s c
```

Slika 2.1: *Susedni elementi u matrici.*

Uputstvo: *Elementi matrice* `m` *susedni elementu* `m[i][j]` *su svi elementi matrice čiji se indeksi, po apsolutnoj vrednosti, razlikuju najviše za jedan. Element matrice može imati najviše osam suseda:* `m[i-1][j-1]`, `m[i-1][j]`, `m[i-1][j+1]`, `m[i][j-1]`, `m[i][j+1]`, `m[i+1][j-1]`, `m[i+1][j]` *i* `m[i+1][j+1]`. *U zavisnosti od položaja u matrici, element matrice može imati i tri ili pet suseda. Na slici 2.1 su slovom* `s` *obeleženi susedni elementi matrice za elemente* `m[2][2]` *(ele-*

ment je na slici obeležen sa `a`*),* `m[0][7]` *(element je na slici obeležen sa* `b`*) i* `m[5][9]` *(element je na slici obeležen sa* `c`*).*

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 4 5
 Unesite elemente matrice:
 1 1 2 1 3
 0 8 1 9 0
 1 1 1 0 0
 0 3 0 2 2
 Indeksi elemenata koji su
 jednaki zbiru suseda su:
 1 1
 3 1
 3 4
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 7 10 12 20
 -1 -3 1 7
 0 -47 2 0
 Indeksi elemenata koji su
 jednaki zbiru suseda su:
 0 3
 1 2
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 5 -3
 Greska: neispravan unos.
```

Zadatak 2.7.7 Napisati funkciju koja formira niz $b_0, b_1, \ldots, b_{n-1}$ od matrice $n \times m$ tako što element niza b_i izračunava kao srednju vrednost elemenata i-te vrste matrice. Napisati program koji za učitanu matricu celih brojeva ispisuje dobijeni niz. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 4 5
 Unesite elemente matrice:
 1 1 2 1 3
 0 8 1 9 0
 1 1 1 0 0
 0 3 0 2 2
 Dobijeni niz je:
 1.6 3.6 0.6 1.4
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 4
 Unesite elemente matrice:
 7 10 12 20
 -1 -3 1 7
 0 -47 2 0
 Dobijeni niz je:
 12.25 1 -11.25
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 51 13
 Greska: neispravan unos.
```

Zadatak 2.7.8 Relacija se može predstaviti kvadratnom matricom nula i jedinica na sledeći način: element i je u relaciji sa elementom j ukoliko se u preseku i–te vrste i j–te kolone nalazi jedinica, a nije u relaciji ukoliko se tu nalazi nula. Napisati funkcije:

(a) `int refleksivna(int a[][MAKS], int n)` kojom se za relaciju zadatu matricom a dimenzije $n \times n$ ispituje da li je refleksivna;

(b) `int simetricna(int a[][MAKS], int n)` kojom se za relaciju zadatu matricom a dimenzije $n \times n$ ispituje da li je simetrična;

(c) `int tranzitivna(int a[][MAKS], int n)` kojom se za relaciju zadatu matricom a ispituje dimenzije $n \times n$ da li je tranzitivna;

(d) `int ekvivalencija(int a[][MAKS], int n)` kojom se za relaciju zadatu matricom a dimenzije $n \times n$ ispituje da li je relacija ekvivalencije.

Napisati program koji za učitanu dimenziju n i kvadratnu matricu dimenzije $n \times n$ ispisuje osobine odgovarajuće relacije. Pretpostaviti da je maksimalna dimenzija matrice 50×50 i da matrica za vrednosti elemenata može imati samo nule i jedinice. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 4
 Unesite elemente matrice:
 1 0 0 0
 0 1 1 0
 0 0 1 0
 0 0 0 0
 Relacija nije refleksivna.
 Relacija nije simatricna.
 Relacija jeste tranzitivna.
 Relacija nije ekvivalencija.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 4
 Unesite elemente matrice:
 1 1 0 0
 1 1 1 0
 0 1 1 0
 0 0 0 1
 Relacija jeste refleksivna.
 Relacija jeste simatricna.
 Relacija nije tranzitivna.
 Relacija nije ekvivalencija.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 54
 Greska: neispravan unos.
```

Zadatak 2.7.9 Data je kvadratna matrica dimenzije $n \times n$.

(a) Napisati funkciju `float trag(float a[][MAKS], int n)` koja računa trag matrice, odnosno zbir elemenata na glavnoj dijagonali matrice.

(b) Napisati funkciju `float suma_sporedna(float a[][MAKS], int n)` koja računa zbir elemenata na sporednoj dijagonali matrice.

(c) Napisati funkciju `float suma_iznad(float a[][MAKS], int n)` koja određuje sumu elemenata iznad glavne dijagonale.

(d) Napisati funkciju `float suma_ispod(float a[][MAKS], int n)` koja određuje sumu elemenata ispod sporedne dijagonale matrice.

Napisati program koji za učitanu matricu realnih brojeva ispisuje na tri decimale trag matrice, sumu na sporednoj dijagonali, sumu iznad glavne dijagonale i sumu elemenata ispod sporedne dijagonale. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 4
 Unesite elemente matrice:
 6 12.08 -1 20.5
 8 90 -33.4 19.02
 7.02 5 -20 14.5
 8.8 -1 3 -22.8
```

Primer 1 (nastavak)

```
Trag: 53.20
Suma na sporednoj dijagonali: 0.90
Suma iznad glavne dijagonale: 31.70
Suma ispod sporedne dijagonale: -7.28
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 5
 Unesite elemente matrice:
 1 2 3 5 5
 7 8 9 0 1
 6 4 3 2 2
 8 9 1 3 4
 0 3 1 8 6
```

Primer 2 (nastavak)

```
Trag: 21.00
Suma na sporednoj dijagonali: 17.00
Suma iznad glavne dijagonale: 33.00
Suma ispod sporedne dijagonale: 31.00
```

Zadatak 2.7.10 Kvadratna matrica je donje trougaona ako se u gornjem trouglu (iznad glavne dijagonale, ne uključujući dijagonalu) nalaze sve nule. Napisati program koji za učitanu kvadratnu matricu proverava da li je ona donje trougaona i ispisuje odgovarajuću poruku. Pretpostaviti da je maksimalna dimenzija matrice 100×100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 5
 Unesite elemente matrice:
 -1 0 0 0 0
 2 10 0 0 0
 0 1 5 0 0
 7 8 20 14 0
 -23 8 5 1 11
 Matrica jeste donje
 trougaona.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 3
 Unesite elemente matrice:
 2 -2 1
 1 2 2
 2 1 -2
 Matrica nije donje
 trougaona.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 200
 Greska: neispravan unos.
```

Zadatak 2.7.11 Napisati program koji za učitanu celobrojnu kvadratnu matricu ispisuje redni broj kolone koja ima najveći zbir elemenata. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 3
 Unesite elemente matrice:
 1 2 3
 7 3 4
 5 3 1
 Indeks kolone je: 0
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 4
 Unesite elemente matrice:
 7 8 9 10
 7 6 11 4
 3 1 2 -2
 8 3 9 9
 Indeks kolone je: 2
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 104
 Greska: neispravan unos.
```

Zadatak 2.7.12 Napisati program koji za učitanu kvadratnu matricu realnih brojeva izračunava i ispisuje na dve decimale razliku između zbira elemenata

gornjeg trougla i zbira elemenata donjeg trougla matrice. Gornji trougao čine svi elementi matrice koji su iznad glavne i sporedne dijagonale (ne računajući dijagonale), a donji trougao čine svi elementi ispod glavne i sporedne dijagonale (ne računajući dijagonale). Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 3
 Unesite elemente matrice:
 2 3.2 4
 7 8.8 1
 2.3 1 1
 Razlika je: 2.20
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 5
 Unesite elemente matrice:
 2.3 1 12 8 -20
 4 -8.2 7 14.5 19
 1 -2.5 9 11 33
 3 4.3 -5.7 2 8
 9 56 1.08 7 5.5
 Razlika je: -30.38
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 52
 Greska: neispravan unos.
```

Zadatak 2.7.13 Napisati program koji za učitanu celobrojnu matricu dimenzije $m \times n$ i uneta dva broja p i k ($p \leq m$, $k \leq n$) ispisuje sume svih podmatrica dimenzije $p \times k$ unete matrice. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Unesite dva cela broja: 3 3
 Sume podmatrica su: 54 63
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice:
 3 4
 Unesite elemente matrice:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Unesite dva cela broja: 2 3
 Sume podmatrica su: 24 30 48 54
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice:
 5 3
 Unesite elemente matrice:
 1 1 2
 5 0 2
 7 8 9
 1 2 4
 0 1 1
 Unesite dva cela broja: 2 2
 Sume podmatrica su: 7 5 20 19 18 23 4 8
```

Primer 4

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice:
 -3 200
 Greska: neispravan unos.
```

Zadatak 2.7.14 Napisati program koji za učitanu celobrojnu kvadratnu matricu ispituje da li su njeni elementi po kolonama, vrstama i dijagonalama (glavnoj i sporednoj) sortirani strogo rastuće. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću

poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 2
 Unesite elemente matrice:
 6 9
 4 10
 Elementi nisu sortirani po kolonama.
 Elementi su sortirani po vrstama.
 Elementi nisu sortirani po dijagonalama.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 7 8 9
 Elementi su sortirani po kolonama.
 Elementi su sortirani po vrstama.
 Elementi su sortirani po dijagonalama.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 4
 Unesite elemente matrice:
 5 5 7 9
 6 10 11 13
 8 12 14 15
 13 15 16 20
 Elementi su sortirani po kolonama.
 Elementi nisu sortirani po vrstama.
 Elementi su sortirani po dijagonalama.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 1
 Unesite elemente matrice:
 5
 Elementi su sortirani po kolonama.
 Elementi su sortirani po vrstama.
 Elementi su sortirani po dijagonalama.
```

Zadatak 2.7.15 Napisati program koji za učitanu celobrojnu kvadratnu matricu ispituje da li su zbirovi elemenata njenih kolona uređeni u strogo rastućem poretku. Pretpostaviti da je maksimalna dimenzija matrice 10×10. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 4
 Unesite elemente matrice:
 1 0 0 0
 0 0 1 0
 0 0 0 1
 0 1 0 0
 Sume nisu uredjene strogo rastuce.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 7 8 9
 Sume jesu uredjene strogo rastuce.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 2 -2 1
 1 2 2
 2 1 -2
 Sume nisu uredjene strogo rastuce.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj vrsta matrice: 5
 Unesite elemente matrice:
 -1 0 3 0 20
 0 0 0 10 0
 0 0 -1 0 0
 0 1 0 0 0
 0 0 0 0 -1
 Sume jesu uredjene strogo rastuce.
```

Zadatak 2.7.16 Matrica je *ortonormirana* ako je vrednost skalarnog proizvod svakog para različitih vrsta jednak nuli, a vrednost skalarnog proizvoda vrste

sa samom sobom jednak jedinici. Napisati program koji za unetu celobrojnu kvadratnu matricu proverava da li je ortonormirana. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. NAPOMENA: *Skalarni proizvod vektora* $a = (a_1, a_2, \ldots, a_n)$ *i* $b = (b_1, b_2, \ldots, b_n)$ *je* $a_1 \cdot b_1 + a_2 \cdot b_2 + \ldots + a_n \cdot b_n$.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 4
 Unesite elemente matrice:
 1 0 0 0
 0 0 1 0
 0 0 0 1
 0 1 0 0
 Matrica jeste ortonormirana.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 7 8 9
 Matrica nije ortonormirana.
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 2 -2 1
 1 2 2
 2 1 -2
 Matrica nije ortonormirana.
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 5
 Unesite elemente matrice:
 -1 0 0 0 0
 0 0 0 1 0
 0 0 -1 0 0
 0 1 0 0 0
 0 0 0 0 -1
 Matrica jeste ortonormirana.
```

Zadatak 2.7.17 Kvadratna matrica je *magični kvadrat* ako su sume elemenata u svim vrstama i kolonama jednake. Napisati program koji proverava da li je data celobrojna kvadratna matrica magični kvadrat i ispisuje odgovarajuću poruku na standardni izlaz. Pretpostaviti da je maksimalna dimenzija matrice 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 4
 Unesite elemente matrice:
 1 5 3 1
 2 1 2 5
 3 2 2 3
 4 2 3 1
 Matrica jeste magicni kvadrat.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 -1 3 3
 Matrica nije magicni kvadrat.
```

*** Zadatak 2.7.18** Napisati program koji učitava celobrojnu kvadratnu matricu i ispisuje elemente matrice u grupama koje su paralelne sa njenom sporednom dijagonalom, počevši od gornjeg levog ugla. Pretpostaviti da je maksimalna dimenzija matrice 100×100. U slučaju neispravnog unosa, ispisati odgovarajuću

poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 7 8 9
 Ispis je:
 1
 2 4
 3 5 7
 6 8
 9
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 5
 Unesite elemente matrice:
 7 -8 1 2 3
 90 11 0 5 4
 12 -9 14 23 8
 80 6 88 17 62
 -22 10 44 57 -200
 Ispis je:
 7
 -8 90
 1 11 12
 2 0 -9 80
 3 5 14 6 -22
 4 23 88 10
 8 17 44
 62 57
 -200
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta matrice:
 -5
 Greska: neispravan unos.
```

*** Zadatak 2.7.19** Napisati funkciju `void mnozenje(int a[][MAKS], int m, int n, int b[][MAKS], int k, int t, int c[][MAKS])` koja računa matricu c kao proizvod matrica a i b. Dimenzija matrice a je $n \times m$, a dimenzija matrice b je $k \times t$. Napisati program koji ispisuje proizvod učitanih matrica. Pretpostaviti da je maksimalna dimenzija matrica 50×50. Ukoliko množenje matrica nije moguće ili je došlo do greške prilikom unosa podataka, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice a:
 3 4
 Unesite elemente matrice:
 1 2 8 9
 -4 5 2 3
 7 6 4 10
 Unesite broj vrsta i broj kolona matrice b:
 4 2
 Unesite elemente matrice:
 11 5
 6 7
 8 9
 0 -3
 Rezultat mnozenja je:
 87 64
 2 24
 145 83
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice a:
 5 2
 Unesite elemente matrice:
 1 7
 9 0
 -10 2
 92 3
 14 -8
 Unesite broj vrsta i broj kolona matrice b:
 2 4
 Unesite elemente matrice:
 7 8 9 10
 -11 2 34 78
 Rezultat mnozenja je:
 -70 22 247 556
 63 72 81 90
 -92 -76 -22 56
 611 742 930 1154
 186 96 -146 -484
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice a:
 3 4
 Unesite elemente matrice:
 1 2 8 9
 -4 5 2 3
 7 6 4 10
 Unesite broj vrsta i broj kolona matrice b:
 5 2
 Mnozenje matrica nije moguce.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice a:
 -3 4
 Greska: neispravan unos.
```

*** Zadatak 2.7.20** Element matrice naziva se *sedlo* ako je istovremeno najmanji u svojoj vrsti, a najveći u svojoj koloni. Napisati program koji ispisuje indekse i vrednosti onih elemenata matrice realnih brojeva koji su sedlo. Maksimalna dimenzija matrice je 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 2 3
 Unesite elemente matrice:
 1 2 3
 0 5 6
 Sedlo: 0 0 1
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 3
 Unesite elemente matrice:
 10 3 20
 15 5 100
 30 -1 200
 Sedlo: 1 1 5
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 -3
 Greska: neispravan unos.
```

*** Zadatak 2.7.21** Napisati program koji ispisuje elemente matrice celih brojeva u spiralnom redosledu počevši od gornjeg levog ugla krećući se u smeru kazaljke na satu. Maksimalna dimenzija matrice je 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 3 3
 Unesite elemente matrice:
 1 2 3
 4 5 6
 7 8 9
 Ispis je:
 1 2 3 6 9 8 7 4 5
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i
 broj kolona matrice:
 5 7
 Unesite elemente matrice:
 7 -8 1 2 3 -54 87
 90 11 0 5 4 9 18
 12 -9 14 23 8 -22 74
 80 6 88 17 62 38 41
 -22 10 44 57 -200 39 55
 Ispis je:
 7 -8 1 2 3 -54 87 18 74 41 55
 39 -200 57 44 10 -22 80 12 90
 11 0 5 4 9 -22 38 62 17 88 6
 -9 14 23 8
```

*** Zadatak 2.7.22** Matrica a se sadrži u matrici b ukoliko postoji podmatrica matrice b identična matrici a. Napisati program koji za dve učitane matrice celih brojeva proverava da li se druga matrica sadrži u prvoj učitanoj matrici. Maksimalna dimenzija matrica je 50×50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice A:
 3 4
 Unesite elemente matrice:
 1 2 8 9
 -4 5 2 3
 7 6 4 10
 Unesite broj vrsta i broj kolona matrice B:
 2 2
 Unesite elemente matrice:
 2 3
 4 10
 Druga matrica je sadrzana u prvoj matrici.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice A:
 3 4
 Unesite elemente matrice:
 1 2 8 9
 -4 5 2 3
 7 6 4 10
 Unesite broj vrsta i broj kolona matrice B:
 2 2
 Unesite elemente matrice:
 2 8
 6 4
 Druga matrica nije sadrzana
 u prvoj matrici.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice A:
 5 5
 Unesite elemente matrice:
 7 -8 1 2 3
 90 11 0 5 4
 12 -9 14 23 8
 80 6 88 17 62
 -22 10 44 57 -200
 Unesite broj vrsta i broj kolona matrice B:
 3 4
 Unesite elemente matrice:
 90 11 0 5
 12 -9 14 23
 80 6 88 17
 Druga matrica je sadrzana u prvoj matrici.
```

Primer 4

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona matrice A:
 5 5
 Unesite elemente matrice:
 7 -8 1 2 3
 90 11 0 5 4
 12 -9 14 23 8
 80 6 88 17 62
 -22 10 44 57 -200
 Unesite broj vrsta i broj kolona matrice B:
 53 4
 Greska: neispravan unos.
```

2.8 Rešenja

Rešenje 2.7.1

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int i, j, m, n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);

  /* Ispis elemenata matrice. */
  printf("Matrica je:\n");
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
  }

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.2

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAKS 50

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
```

```
  int i, j, m, n, suma = 0;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);

  /* Racunanje sume kvadrata svih elemenata. */
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      suma += a[i][j] * a[i][j];

  /* Ispis rezultata. */
  printf("Euklidska norma: %.3lf\n", sqrt(suma));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.3

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija ispisuje elemente matrice dimenzije mxn. */
void ispisi(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Matrica je:\n");
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
```

```
  }
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int m, n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Ispis ucitane matrice. */
  ispisi(a, m, n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.4

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija ispisuje elemente matrice dimenzije mxn. */
void ispisi(int a[][MAKS], int m, int n) {
  int i, j;
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
  }
}
```

```
/* Funkcija formira maticu t transponovanjem matrice a. */
void transponovana(int a[][MAKS], int m, int n, int t[][MAKS]) {
  int i, j;
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      t[j][i] = a[i][j];
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS], t[MAKS][MAKS];
  int m, n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Formiranje transponovane matrice. */
  transponovana(a, m, n, t);

  /* Ispis rezultata. */
  printf("Transponovana matrica je:\n");
  ispisi(t, n, m);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.5

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija ispisuje elemente matrice dimenzije mxn. */
```

```
void ispisi(int a[][MAKS], int m, int n) {
  int i, j;
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
  }
}

/* Funkcija razmenjuje elemente k-te i t-te vrste. */
void razmeni(int a[][MAKS], int m, int n, int k, int t) {
  int j, pom;
  for (j = 0; j < n; j++) {
    pom = a[k][j];
    a[k][j] = a[t][j];
    a[t][j] = pom;
  }
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int m, n, k, t;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Ucitavanje indeksa vrsta i provera ispravnosti ulaza. */
  printf("Unesite indekse vrsta: ");
  scanf("%d%d", &k, &t);
  if (k < 0 || k >= m || t < 0 || t >= m) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Razmena k-te i t-te vrste. */
  razmeni(a, m, n, k, t);

  /* Ispis rezultata. */
  printf("Rezultujuca matrica:\n");
  ispisi(a, m, n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.6

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int m, n, i, j, suma_suseda;
  int k, t;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Izracunavanje i ispis rezultata. */
  printf("Indeksi elemenata koji su jednaki zbiru suseda su:\n");
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++) {
      suma_suseda = 0;

      /* Racunanje sume elemenata podmatrice velicine 3*3 ciji je
         centralni element a[i][j]. Pri racunanju ove sume vodi se
         racuna da se ne izadje iz okvira matice a. Preciznije,
         ukoliko su sracunate neodgovarajuce vrednosti za indekse
         k i t (npr. kada je i=0 ili kada je i=m-1), te vrednosti
         zahvaljuci uslovu k >= 0 && k < m && t >= 0 && t < n nece
         uci u sumu. */
      for (k = i - 1; k <= i + 1; k++)
        for (t = j - 1; t <= j + 1; t++)
          if (k >= 0 && k < m && t >= 0 && t < n)
            suma_suseda += a[k][t];

      /* Od ukupne sume se oduzima tekuci element kako bi se dobio
```

```
51          zbir elemenata koji su njegovi susedi. */
      suma_suseda -= a[i][j];
53
      /* Ukoliko je suma suseda jednaka tekucem elementu, ispisuju
55       se indeksi tekuceg elementa matrice. */
      if (suma_suseda == a[i][j])
57      printf("%d %d\n", i, j);
    }
59  }
  exit(EXIT_SUCCESS);
61}
```

Rešenje 2.7.7

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  #define MAKS 50
5
  /* Funkcija ucitava elemente matrice dimenzije mxn. */
7 void ucitaj(int a[][MAKS], int m, int n) {
    int i, j;
9
    printf("Unesite elemente matrice:\n");
11  for (i = 0; i < m; i++)
      for (j = 0; j < n; j++)
13      scanf("%d", &a[i][j]);
  }
15
  /* Funkcija formira niz b tako sto element b[i] ima vrednost
17    prosecne vrednosti i-te vrste matrice. */
  void kreiraj_niz(int a[][MAKS], int m, int n, double b[]) {
19  int i, j, suma;

21  for (i = 0; i < m; i++) {
      suma = 0;
23    for (j = 0; j < n; j++)
        suma += a[i][j];
25
      b[i] = (double) suma / n;
27  }
  }
29
  int main() {
31  /* Deklaracije potrebnih promenljivih. */
    int a[MAKS][MAKS];
33  double b[MAKS];
    int m, n, i;
35
    /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
37  printf("Unesite broj vrsta i broj kolona matrice: ");
```

```
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Formiranje niza b. */
  kreiraj_niz(a, m, n, b);

  /* Ispis rezultata. */
  printf("Dobijeni niz je:\n");
  for (i = 0; i < m; i++)
    printf("%g ", b[i]);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.8

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Relacija je refleksivna ukoliko je za svako i a[i][i] = 1.
   Funkcija proverava da li je relacija zadata matricom a
   refleksivna i vraca 1 ukoliko jeste, a 0 inace. */
int refleksivna(int a[][MAKS], int n) {
  int i;
  for (i = 0; i < n; i++)
    if (a[i][i] != 1)
      return 0;

  return 1;
}

/* Relacija je simetricna ukoliko za svaki par i, j vazi da je
   a[i][j] = a[j][i]. Funkcija proverava da li je relacija zadata
```

```
   matricom a simetricna i vraca 1 ukoliko jeste, a 0 inace. */
int simetricna(int a[][MAKS], int n) {
  int i, j;
  for (i = 0; i < n; i++)
    for (j = i + 1; j < n; j++)
      if (a[i][j] != a[j][i])
        return 0;

  return 1;
}

/* Relacija je tranzitivna ukoliko za svaku trojku i, j, k vazi da
   ako je a[i][j] = 1 i a[j][k] = 1, onda je i a[i][k] = 1.
   Funkcija proverava da li je relacija zadata matricom a
   tranzitivna i vraca 1 ukoliko jeste, a 0 inace. */
int tranzitivna(int a[][MAKS], int n) {
  int i, j, k;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      for (k = 0; k < n; k++)
        if (a[i][j] == 1 && a[j][k] == 1 && a[i][k] == 0)
          return 0;

  return 1;
}

/* Relacija je relacija ekvivalencije ukoliko je refleksivna,
   simetricna i tranzitivna. Funkcija proverava da li je relacija
   zadata matricom a relacija ekvivalencije i vraca 1 ukoliko
   jeste, a 0 inace. */
int ekvivalencija(int a[][MAKS], int n) {
  if (refleksivna(a, n) && simetricna(a, n) && tranzitivna(a, n))
    return 1;

  return 0;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);
```

```
82    /* Racunanje i ispis rezultata. */
      if (refleksivna(a, n))
84      printf("Relacija jeste refleksivna.\n");
      else
86      printf("Relacija nije refleksivna.\n");

88    if (simetricna(a, n))
        printf("Relacija jeste simetricna.\n");
90    else
        printf("Relacija nije simatricna.\n");
92
      if (tranzitivna(a, n))
94      printf("Relacija jeste tranzitivna.\n");
      else
96      printf("Relacija nije tranzitivna.\n");

98    if (ekvivalencija(a, n))
        printf("Relacija jeste ekvivalencija.\n");
100   else
        printf("Relacija nije ekvivalencija.\n");
102
      exit(EXIT_SUCCESS);
104 }
```

Rešenje 2.7.9

```
  #include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS 50

6 /* Funkcija ucitava elemente matrice dimenzije mxn. */
  void ucitaj(float a[][MAKS], int n) {
8   int i, j;
    printf("Unesite elemente matrice:\n");
10  for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
12      scanf("%f", &a[i][j]);
  }
14
  /* Funkcija racuna trag matrice. */
16 float trag(float a[][MAKS], int n) {
    float suma = 0;
18  int i;

20  for (i = 0; i < n; i++)
      suma += a[i][i];
22
    return suma;
24 }
```

```
26 /* Funkcija racuna sumu elemenata koji se nalaze na sporednoj
      dijagonali matrice. */
28 float suma_sporedna(float a[][MAKS], int n) {
     float suma = 0;
30   int i;

32   for (i = 0; i < n; i++)
       suma += a[i][n - i - 1];
34
     return suma;
36 }

38 /* Funkcija racuna sumu elemenata koji se nalaze iznad glavne
      dijagonale matrice. */
40 float suma_iznad(float a[][MAKS], int n) {
     float suma = 0;
42   int i, j;

44   for (i = 0; i < n; i++)
       for (j = i + 1; j < n; j++)
46       suma += a[i][j];

48   return suma;
   }
50
   /* Funkcija racuna sumu elemenata koji se nalaze ispod sporedne
52    dijagonale matrice. */
   float suma_ispod(float a[][MAKS], int n) {
54   float suma = 0;
     int i, j;
56
     for (i = 0; i < n; i++)
58     for (j = n - i; j < n; j++)
         suma += a[i][j];
60
     return suma;
62 }

64 int main() {
     /* Deklaracije potrebnih promenljivih. */
66   float a[MAKS][MAKS];
     int n;
68
     /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
70   printf("Unesite broj vrsta matrice: ");
     scanf("%d", &n);
72   if (n <= 0 || n > MAKS) {
       printf("Greska: neispravan unos.\n");
74     exit(EXIT_FAILURE);
     }
76
```

```
  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Ispis rezultata. */
  printf("Trag: %.2f\n", trag(a, n));
  printf("Suma na sporednoj dijagonali: %.2f\n",
         suma_sporedna(a, n));
  printf("Suma iznad glavne dijagonale: %.2f\n",
         suma_iznad(a, n));
  printf("Suma ispod sporedne dijagonale: %.2f\n",
         suma_ispod(a, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.10

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija proverava da li je matrica donje trougaona i vraca
   jedinicu ukoliko jeste, a nulu inace. */
int donje_trougaona(int a[][MAKS], int n) {
  int i, j;

  /* Prolazi se kroz sve elemente iznad glavne dijagonale i ukoliko
     se naidje na element koji je razlicit od nule, onda matrica
     nije donje trougaona. */
  for (i = 0; i < n; i++)
    for (j = i + 1; j < n; j++)
      if (a[i][j] != 0)
        return 0;

  /* Ukoliko su svi elementi iznad glavne dijagonale nule, matrica
     jeste donje trougaona. */
  return 1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
```

```
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Ispis rezultata. */
  if (donje_trougaona(a, n))
    printf("Matrica jeste donje trougaona.\n");
  else
    printf("Matrica nije donje trougaona.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.11

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n, i, j;
  int maksimalni_zbir, trenutni_zbir = 0, indeks_kolone;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
```

```
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Maksimalni zbir se inicijalizuje na vrednost zbira prve
     kolone. U ovom slucaju bi bilo pogresno da se maksimalni zbir
     inicijalizuje na nulu jer moze da se desi da su svi elementi
     matrice negativni. Drugi nacin da se ispravno inicijalizuje
     maksimalni zbir jeste da mu se dodeli vrednost konstante
     INT_MIN cija se definicija nalazi u zaglavlju limits.h. */
  for (i = 0; i < n; i++)
    trenutni_zbir += a[i][0];

  maksimalni_zbir = trenutni_zbir;
  indeks_kolone = 0;

  /* Racunanje zbira svake sledece kolone i azuriranje vrednosti
     maksimalnog zbira. */
  for (j = 1; j < n; j++) {
    /* Racunanje zbira kolone j. */
    trenutni_zbir = 0;
    for (i = 0; i < n; i++)
      trenutni_zbir += a[i][j];

    /* Ukoliko je taj zbir veci od trenutno maksimalnog zbira,
       azurira se vrednost maksimalnog zbira i pamti se tekuca
       kolona. */
    if (trenutni_zbir > maksimalni_zbir) {
      maksimalni_zbir = trenutni_zbir;
      indeks_kolone = j;
    }
  }

  /* Ispis rezultata. */
  printf("Indeks kolone je: %d\n", indeks_kolone);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.12

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(float a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
```

```
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%f", &a[i][j]);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  float a[MAKS][MAKS];
  int n, i, j;
  float gornji_trougao = 0, donji_trougao = 0;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Racunanje sume gornjeg trougla. */
  for (i = 0; i < n / 2; i++)
    for (j = i + 1; j < n - i - 1; j++)
      gornji_trougao += a[i][j];

  /* Racunanje sume donjeg trougla. */
  for (i = n / 2; i < n; i++)
    for (j = n - i; j < i; j++)
      donji_trougao += a[i][j];

  /* Ispis rezultata. */
  printf("Razlika je: %.2f\n", gornji_trougao - donji_trougao);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.13

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
```

```
12        scanf("%d", &a[i][j]);
  }
14
  int main() {
16  /* Deklaracije potrebnih promenljivih. */
    int a[MAKS][MAKS];
18  int n, i, j, m, x, y, p, k;
    int suma;
20
    /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
22  printf("Unesite broj vrsta i broj kolona matrice: ");
    scanf("%d%d", &m, &n);
24  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
      printf("Greska: neispravan unos.\n");
26    exit(EXIT_FAILURE);
    }
28
    /* Ucitavanje elemenata matrice. */
30  ucitaj(a, m, n);

32  /* Ucitavanje brojeva p i k i provera ispravnosti ulaza. */
    printf("Unesite dva cela broja: ");
34  scanf("%d%d", &p, &k);
    if (p <= 0 || p > m || k <= 0 || k > n) {
36    printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
38  }

40  /* Racunanje i ispis rezultata. */
    printf("Sume podmatrica su: ");
42  for (i = 0; i <= m - p; i++) {
      for (j = 0; j <= n - k; j++) {
44      /* Za svaku poziciju (i,j), racunana se suma podmatrice
           dimenzije pxk, ciji je gornji levi ugao a[i][j]. */
46      suma = 0;
        for (x = 0; x < p; x++)
48        for (y = 0; y < k; y++)
            suma += a[i + x][j + y];
50
        printf("%d ", suma);
52    }
    }
54  printf("\n");

56  exit(EXIT_SUCCESS);
  }
```

Rešenje 2.7.14

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija proverava da li je kolona j sortirana rastuce i vraca
   jedinicu ukoliko jeste, a nulu inace. */
int sortirana_kolona(int a[][MAKS], int n, int j) {
  int i;

  for (i = 0; i < n - 1; i++)
    if (a[i][j] >= a[i + 1][j])
      return 0;

  return 1;
}

/* Funkcija proverava da li je svaka kolona matrice sortirana
   rastuce i vraca jedinicu ukoliko jeste, a nulu inace. */
int sortirani_po_kolonama(int a[][MAKS], int n) {
  int j;

  for (j = 0; j < n; j++)
    if (!sortirana_kolona(a, n, j))
      return 0;

  return 1;
}

/* Funkcija proverava da li je i-ta vrsta sortirana rastuce i vraca
   jedinicu ukoliko jeste, a nulu inace. */
int sortirana_vrsta(int a[][MAKS], int n, int i) {
  int j;

  for (j = 0; j < n - 1; j++)
    if (a[i][j] >= a[i][j + 1])
      return 0;

  return 1;
}
```

```
/* Funkcija proverava da li je svaka vrsta matrice sortirana
   rastuce i vraca jedinicu ukoliko jeste, a nulu inace. */
int sortirani_po_vrstama(int a[][MAKS], int n) {
  int i;

  for (i = 0; i < n; i++)
    if (!sortirana_vrsta(a, n, i))
      return 0;

  return 1;
}

/* Funkcija proverava da li je glavna dijagonala matrice sortirana
   rastuce i vraca jedinicu ukoliko jeste, a nulu inace. */
int sortirana_glavna(int a[][MAKS], int n) {
  int i;

  for (i = 0; i < n - 1; i++)
    if (a[i][i] >= a[i + 1][i + 1])
      return 0;

  return 1;
}

/* Funkcija proverava da li je sporedna dijagonala matrice
   sortirana rastuce i vraca jedinicu ukoliko jeste, a nulu inace. */
int sortirana_sporedna(int a[][MAKS], int n) {
  int i;

  for (i = 0; i < n - 1; i++)
    if (a[i][n - i - 1] >= a[i + 1][n - i - 2])
      return 0;

  return 1;
}

/* Funkcija proverava da li su obe dijagonale matrice sortirane
   rastuce i vraca jedinicu ukoliko jesu, a nulu inace. */
int sortirani_po_dijagonalama(int a[][MAKS], int n) {
  return sortirana_glavna(a, n) && sortirana_sporedna(a, n);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
```

```
103     exit(EXIT_FAILURE);
    }
105
    /* Ucitavanje elemenata matrice. */
107 ucitaj(a, n);

109 /* Ispis rezultata. */
    if (sortirani_po_kolonama(a, n))
111   printf("Elementi su sortirani po kolonama.\n");
    else
113   printf("Elementi nisu sortirani po kolonama.\n");

115 if (sortirani_po_vrstama(a, n))
      printf("Elementi su sortirani po vrstama.\n");
117 else
      printf("Elementi nisu sortirani po vrstama.\n");
119
    if (sortirani_po_dijagonalama(a, n))
121   printf("Elementi su sortirani po dijagonalama.\n");
    else
123   printf("Elementi nisu sortirani po dijagonalama.\n");

125 exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.15

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  #define MAKS 10
5
  /* Funkcija ucitava elemente matrice dimenzije mxn. */
7 void ucitaj(int a[][MAKS], int n) {
    int i, j;
9   printf("Unesite elemente matrice:\n");
    for (i = 0; i < n; i++)
11    for (j = 0; j < n; j++)
        scanf("%d", &a[i][j]);
13 }

15 /* Funkcija racuna sumu elemenata kolone j. */
  int suma_kolone(int a[][MAKS], int n, int j) {
17   int suma = 0, i;

19   for (i = 0; i < n; i++)
       suma += a[i][j];
21
     return suma;
23 }
```

```
/* Funkcija proverava da li su sume kolona uredjene rastuce i vraca
   jedinicu ako jesu, a nulu inace. */
int uredjene_sume(int a[][MAKS], int n) {
  int prethodna_suma, trenutna_suma, j;

  /* Prva suma se inicijalizuje na sumu prve kolone. */
  prethodna_suma = suma_kolone(a, n, 0);

  for (j = 1; j < n; j++) {
    /* Racunanje sume trenutne kolone. */
    trenutna_suma = suma_kolone(a, n, j);

    /* Ukoliko je ta suma manja ili jednaka prethodnoj, poredak
       suma nije rastuci. */
    if (trenutna_suma <= prethodna_suma)
      return 0;

    /* Suma trenutne kolone postaje suma prethodne kolone za
       narednu iteraciju. */
    prethodna_suma = trenutna_suma;
  }

  return 1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Ispis rezultata. */
  if (uredjene_sume(a, n))
    printf("Sume jesu uredjene strogo rastuce.\n");
  else
    printf("Sume nisu uredjene strogo rastuce.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.16

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 200

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija racuna skalarni proizvod i-te i j-te vrste matrice. */
int skalarni_proizvod(int a[][MAKS], int n, int i, int j) {
  int suma = 0, k;

  for (k = 0; k < n; k++)
    suma += a[i][k] * a[j][k];

  return suma;
}

/* Matrica je ortonormirana ukoliko je skalarni proizvod svakog
   para razlicitih vrsta jednak nuli, a skalarni proizvod svake
   vrste same sa sobom jednak jedinici. Funkcija proverava da li je
   matrica ortorormirana i vraca jedinicu ukoliko jeste, a nulu
   inace. */
int ortonormirana(int a[][MAKS], int n) {
  int i, j;

  /* Za svaki par vrsta se racuna skalarni proizvod i proverava da
     li je uslov ispunjen. Ukoliko nije, kao povratna vrednost
     funkcije se vraca nula. */
  for (i = 0; i < n; i++)
    for (j = i; j < n; j++) {
      /* Provera za slucaj kada se racuna skalarni proizvod vrste
         same sa sobom. */
      if (i == j && skalarni_proizvod(a, n, i, i) != 1)
        return 0;

      /* Provera za par razlicitih vrsta. */
      if (i != j && skalarni_proizvod(a, n, i, j) != 0)
        return 0;
    }

  /* Ako je izvrsavanje stiglo do kraja petlje, znaci da je uslov
     ispunjen za sve vrste, tj. da je matrica ortonormirana. */
  return 1;
```

```
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);

  /* Ispis rezultata. */
  if (ortonormirana(a, n))
    printf("Matrica jeste ortonormirana.\n");
  else
    printf("Matrica nije ortonormirana.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.17

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija racuna sumu kolone j. */
int suma_kolone(int a[][MAKS], int n, int j) {
  int i, suma = 0;

  for (i = 0; i < n; i++)
    suma += a[i][j];

  return suma;
```

```
}

/* Funkcija racuna sumu i-te vrste. */
int suma_vrste(int a[][MAKS], int n, int i) {
  int j, suma = 0;

  for (j = 0; j < n; j++)
    suma += a[i][j];

  return suma;
}

/* Funkcija proverava da li elementi matrice predstavljaju magicni
   kvadrat. */
int magicni_kvadrat(int a[][MAKS], int n) {
  /* Da bi matrica bila magicni kvadrat, sume svih vrsta i kolona
     treba da budu jednke. Suma se zato inicijalizuje na sumu prve
     kolone. */
  int suma = suma_kolone(a, n, 0);
  int i, j;

  /* Proverava se da li su sume ostalih kolona jednake izracunatoj
     sumi. Ukoliko se naidje na kolonu koja ne zadovoljava ovaj
     uslov, matrica nije magicni kvadrat. */
  for (j = 1; j < n; j++)
    if (suma_kolone(a, n, j) != suma)
      return 0;

  /* Proverava se i da li su sume svih vrsta jednake izracunatoj
     sumi. Ukoliko se naidje na vrstu koja ne zadovoljava ovaj
     uslov, matrica nije magicni kvadrat. */
  for (i = 0; i < n; i++)
    if (suma_vrste(a, n, i) != suma)
      return 0;

  /* Ako sve vrste i kolone imaju jednake sume, matrica je magicni
     kvadrat. */
  return 1;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int n;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }
```

```
75
  /* Ucitavanje elemenata matrice. */
77  ucitaj(a, n);

79  /* Ispis rezultata. */
  if (magicni_kvadrat(a, n))
81    printf("Matrica jeste magicni kvadrat.\n");
  else
83    printf("Matrica nije magicni kvadrat.\n");

85  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.18

```
#include <stdio.h>
2 #include <stdlib.h>

4 #define MAKS 100

6 /* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int n) {
8   int i, j;
  printf("Unesite elemente matrice:\n");
10  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
12      scanf("%d", &a[i][j]);
}
14
int main() {
16  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
18  int n, i, j, k;

20  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta matrice: ");
22  scanf("%d", &n);
  if (n <= 0 || n > MAKS) {
24    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
26  }

28  /* Ucitavanje elemenata matrice. */
  ucitaj(a, n);
30
  /* Petlja kojom se ispisuju dijagonale iznad sporedne dijagonale,
32     ukljucujuci i sporednu dijagonalu.
     Npr. za n=4, indeksi elemenata u matrici su:
34     (0,0) (0,1) (0,2) (0,3)
     (1,0) (1,1) (1,2) (1,3)
36     (2,0) (2,1) (2,2) (2,3)
```

```
     (3,0) (3,1) (3,2) (3,3)
     Dakle, ispis elemenata ide u sledecem redosledu:
     (0,0)
     (0,1) (1,0)
     (0,2) (1,1) (2,0)
     (0,3) (1,2) (2,1) (3,0)
     Za k-ti ispis vazi da indeksi vrsta imaju vrednosti od nula do
     k, a indeksi kolona od k do nula. */
  printf("Ispis je:\n");
  for (k = 0; k < n; k++) {
    /* Indeks kolone se inicijalizuje na k, a indeks vrste na 0. */
    j = k;
    i = 0;

    /* Ispisuju se odgovarajuci elementi, indeks vrste se povecava,
       a indeks kolone se smanjuje. */
    while (j >= 0) {
      printf("%d ", a[i][j]);
      i++;
      j--;
    }
    printf("\n");
  }

  /* Petlja kojom se ispisuju dijagonale ispod sporedne dijagonale.
     Npr. za n=4, indeksi elemenata u matrici su:
     (0,0) (0,1) (0,2) (0,3)
     (1,0) (1,1) (1,2) (1,3)
     (2,0) (2,1) (2,2) (2,3)
     (3,0) (3,1) (3,2) (3,3)
     Dakle, ispis elemenata ide u sledecem redosledu:
     (1,3) (2,2) (3,1)
     (2,3) (3,2)
     (3,3)
     Za k-ti ispis vazi da indeksi vrsta imaju vrednosti od k do
     n-1, a indeksi kolona od n-1 do 1. */
  for (k = 1; k < n; k++) {
    i = k;
    j = n - 1;

    while (i < n) {
      printf("%d ", a[i][j]);
      i++;
      j--;
    }
    printf("\n");
  }

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.19

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija ispisuje elemente matrice dimenzije mxn. */
void ispisi(int a[][MAKS], int m, int n) {
  int i, j;
  for (i = 0; i < m; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", a[i][j]);
    printf("\n");
  }
}

/* Funkcija vrsi mnozenje matrica a i b i rezultat smesta u matricu
   c. */
void mnozenje(int a[][MAKS], int m, int n,
              int b[][MAKS], int k, int t, int c[][MAKS]) {
  int i, j, w;

  for (i = 0; i < m; i++)
    for (j = 0; j < t; j++) {
      /* Element c[i][j] se dobija kao skalarni proizvod i-te vrste
         matrice a i j-te kolone matrice b. */
      c[i][j] = 0;
      for (w = 0; w < n; w++)
        c[i][j] += a[i][w] * b[w][j];
    }
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS], b[MAKS][MAKS], c[MAKS][MAKS];
  int m, n, k, t;

  /* Ucitavanje dimenzija matrice a i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice a: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
```

```
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata prve matrice. */
  ucitaj(a, m, n);

  /* Ucitavanje dimenzija matrice b i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice b: ");
  scanf("%d%d", &k, &t);
  if (k <= 0 || k > MAKS || t <= 0 || t > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Provera da li se odgovarajuce dimenzije matrica poklapaju. */
  if (n != k) {
    printf("Mnozenje matrica nije moguce.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata druge matrice. */
  ucitaj(b, k, t);

  /* Racunanje proizvoda. */
  mnozenje(a, m, n, b, k, t, c);

  /* Ispis rezultata. */
  printf("Rezultat mnozenja je:\n");
  ispisi(c, m, t);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.20

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(double a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%lf", &a[i][j]);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  double a[MAKS][MAKS];
```

```
  int m, n, k, i, j, indeks_kolone;
  double maks_kolone, min_vrste;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Pronalazak elemenata koji su sedlo. */
  for (i = 0; i < m; i++) {
    /* Pronalazi se najmanji element u tekucoj vrsti. Pamti se
       kolona kojoj taj element pripada. */
    min_vrste = a[i][0];
    indeks_kolone = 0;

    for (j = 1; j < n; j++)
      if (a[i][j] < min_vrste) {
        min_vrste = a[i][j];
        indeks_kolone = j;
      }

    /* Pronalazi se najveci element u zapamcenoj koloni. */
    maks_kolone = a[0][indeks_kolone];

    for (k = 1; k < m; k++)
      if (a[k][indeks_kolone] > maks_kolone)
        maks_kolone = a[k][indeks_kolone];

    /* Element je sedlo ukoliko je on istovremeno najmanji u svojoj
       vrsti i najveci u svojoj koloni. */
    if (min_vrste == maks_kolone)
      printf("Sedlo: %d %d %g\n", i, indeks_kolone, min_vrste);
  }

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.21

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
```

```
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int a[MAKS][MAKS];
  int m, n, brojac, i, j, pravac;
  int gornja_granica, donja_granica, leva_granica, desna_granica;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  printf("Unesite broj vrsta i broj kolona matrice: ");
  scanf("%d%d", &m, &n);
  if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  ucitaj(a, m, n);

  /* Ciklicni ispis elemenata matrice:
     Npr. za n=4, indeksi elemenata u matrici su:
     (0,0) (0,1) (0,2) (0,3)
     (1,0) (1,1) (1,2) (1,3)
     (2,0) (2,1) (2,2) (2,3)
     (3,0) (3,1) (3,2) (3,3)
     Ispis treba da ide sledecim redosledom:
     1. krece se sa leva na desno (0,0) (0,1) (0,2) (0,3)
     2. zatim se ide na dole (1,3) (2,3) (3,3)
     3. zatim na levo (3,2) (3,1) (3,0)
     4. zatim na gore (2,0) (1,0) (ovde se staje jer je (0,0) vec
     ispisano) i prelazi se opet na levo. Koraci 1-4 se ponavljaju
     dok god se ne ispisu svi elementi. Ideja je da kada se ispisu
     elementi prve vrste (kada se ide sa leva na desno), da se
     pomeri "gornja granica ispisa" za 1, kako bi se naznacilo da
     je taj red vec ispisan. Slicno, kada se vrsi ispis odozgo na
     dole, uspesno je ispisana jedna kolona pa je potrebno pomeriti
     "desnu granicu ispisa" za jedan u levo. Kada se ispise jedna
     vrsta sa desna na levo, vrsi se pomeranje donje granice ispisa
     za jedan na gore. Slicno, kada se ispise jedna kolona odozdo na
     gore, pomera se leva granica ispisa za jedan u desno. */
  gornja_granica = 0;
  donja_granica = m - 1;
  leva_granica = 0;
  desna_granica = n - 1;

  /* Promenljiva pravac govori u kom smeru ispis ide. */
  pravac = 1;
```

```
61  /* Promenljive i i j su indeksi elementa koji se ispisuje. */
    i = 0;
63  j = 0;

65  printf("Ispis je:\n");
    for (brojac = 0; brojac < m * n; brojac++) {
67    printf("%d ", a[i][j]);

69    switch (pravac) {
        /* Ako je pravac = 1, trenutni smer ispisa je sa leva na
71         desno. */
      case 1:
73      /* Ako je ispisan element na desnoj granici, onda se menja
           pravac ispisa. */
75      if (j == desna_granica) {
          /* Prelazi se na pravac odozgo na dole. */
77        pravac = 2;
          /* Pomera se gornja granica za jedan na dole. */
79        gornja_granica++;
          /* Pomera se vrednost vrste za jedan na dole. */
81        i++;
        } else {
83        /* Ako jos uvek nije ispisan element na desnoj granici,
             vrsi se pomeranje na sledeci element u trenutnoj vrsti. */
85        j++;
        }
87      break;

89      /* Ako je pravac = 2, trenutni smer ispisa je odozgo na dole.
           Slicno kao i u prethodnom slucaju, ako se dodje do donje
91         granice, menja se pravac i pomera se desna granica za
           jedno mesto u levo. U suprotnom se samo prelazi na narednu
93         vrstu. */
      case 2:
95      if (i == donja_granica) {
          pravac = 3;
97        desna_granica--;
          j--;
99      } else {
          i++;
101     }
        break;
103
        /* Ako je pravac = 3, trenutni smer ispisa je sa desna na
105        levo. Slicno kao i u prethodnom slucaju, ako se dodje do
           leve granice, menja se pravac i pomera se donja granica za
107        jedno mesto na gore. U suprotnom se samo prelazi na
           narednu kolonu. */
109   case 3:
        if (j == leva_granica) {
111       pravac = 4;
```

```
        donja_granica--;
        i--;
      } else {
        j--;
      }
      break;

      /* Ako je pravac = 4, trenutni smer ispisa je odozdo na gore.
         Slicno kao i u prethodnim slucajevima, ako se dodje do
         gornje granice, menja se pravac i pomera se leva granica
         za jedno mesto u desno. U suprotnom se samo prelazi na
         narednu vrstu. */
    case 4:
      if (i == gornja_granica) {
        pravac = 1;
        leva_granica++;
        j++;
      } else {
        i--;
      }
    }
  }
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.7.22

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 50

/* Funkcija ucitava elemente matrice dimenzije mxn. */
void ucitaj(int a[][MAKS], int m, int n) {
  int i, j;
  printf("Unesite elemente matrice:\n");
  for (i = 0; i < m; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &a[i][j]);
}

/* Funkcija proverava da li je matrica b podmatrica matrice a i
   vraca jedinicu ukoliko jeste, a nulu inace. */
int podmatrica(int a[][MAKS], int m, int n,
               int b[][MAKS], int k, int t) {
  int i, j, x, y;
  int jeste_podmatrica;

  for (i = 0; i <= m - k; i++) {
    for (j = 0; j <= n - t; j++) {
```

```
24        /* Za svaku poziciju (i,j) se proverava da li je podmatrica
             dimenzije k*t ciji je gornji levi ugao a[i][j] jednaka
26           matrici b. */
          jeste_podmatrica = 1;
28        for (x = 0; x < k && jeste_podmatrica; x++)
            for (y = 0; y < t && jeste_podmatrica; y++)
30            if (a[i + x][j + y] != b[x][y])
                jeste_podmatrica = 0;
32
          if (jeste_podmatrica)
34          return 1;
        }
36    }

38    return 0;
  }
40
  int main() {
42    /* Deklaracije potrebnih promenljivih. */
      int a[MAKS][MAKS], b[MAKS][MAKS];
44    int m, n, k, t;

46    /* Ucitavanje dimenzije matrice A i provera ispravnosti ulaza. */
      printf("Unesite broj vrsta i broj kolona matrice A: ");
48    scanf("%d%d", &m, &n);
      if (n <= 0 || n > MAKS || m <= 0 || m > MAKS) {
50      printf("Greska: neispravan unos.\n");
        exit(EXIT_FAILURE);
52    }

54    /* Ucitavanje elemenata prve matrice. */
      ucitaj(a, m, n);
56
      /* Ucitavanje dimenzije matrice B i provera ispravnosti ulaza. */
58    printf("Unesite broj vrsta i broj kolona matrice B: ");
      scanf("%d%d", &k, &t);
60    if (k <= 0 || k > MAKS || t <= 0 || t > MAKS) {
        printf("Greska: neispravan unos.\n");
62      exit(EXIT_FAILURE);
      }
64
      /* Ucitavanje elemenata druge matrice. */
66    ucitaj(b, k, t);

68    /* Ispis rezultata. */
      if (podmatrica(a, m, n, b, k, t))
70      printf("Druga matrica je sadrzana u prvoj matrici.\n");
      else
72      printf("Druga matrica nije sadrzana u prvoj matrici.\n");

74    exit(EXIT_SUCCESS);
  }
```

2.9 Strukture

Zadatak 2.9.1 Definisati strukturu kojom se opisuje kompleksan broj. Napisati funkcije koje izračunavaju zbir, razliku, proizvod i količnik dva kompleksna broja. Napisati program koji za učitana dva kompleksna broja ispisuje vrednost zbira, razlike, proizvoda i količnika. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite realni i imaginarni deo prvog broja: 1 2
 Unesite realni i imaginarni deo drugog broja: -2 3
 Zbir: -1.00+5.00*i
 Razlika: 3.00-1.00*i
 Proizvod: -8.00-1.00*i
 Kolicnik: 0.31-0.54*i
```

Zadatak 2.9.2 Definisati strukturu kojom se opisuje razlomak. Napisati funkcije koje izračunavaju zbir i proizvod dva razlomka. Napisati program koji za uneti ceo broj n i unetih n razlomaka ispisuje njihov ukupan zbir i proizvod. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj razlomaka: 5
 Unesite razlomke:
 1 2
 7 8
 3 4
 5 6
 2 9
 Suma svih razlomaka: 229/72
 Proizvod svih razlomaka: 35/576
```

Primer 2

```
Interakcija sa programom:
 Unesite broj razlomaka: 10
 Unesite razlomke:
 4 3
 12 25
 3 8
 1 3
 8 9
 2 3
 5 6
 -24 50
 7 18
 -7 19
 Suma svih razlomaka: 6089/1368
 Proizvod svih razlomaka: 1568/577125
```

Zadatak 2.9.3 Zimi su prehlade česte i treba unositi više vitamina C. Struktura *Vocka* sadrži ime voćke i količinu vitamina C u miligramima (realan broj). Napisati funkcije:

(a) `int ucitaj(Vocka niz[])` koja učitava voćke sa standardnog ulaza sve do kraja ulaza i kao povratnu vrednost vraća broj učitanih voćki;

(b) `Vocka vocka_sa_najvise_vitamina(Vocka niz[], int n)` koja pronalazi

voćku koja ima najviše vitamina C.

Napisati program koji učitava podatke o voćkama i ispisuje ime voćke sa najviše vitamina C. Pretpostaviti da broj voćki neće biti veći od 50, kao i da je ime voćke niska od najviše 20 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite ime vocke i njenu kolicinu vitamina C: jabuka 4.6
 Unesite ime vocke i njenu kolicinu vitamina C: limun 83.5
 Unesite ime vocke i njenu kolicinu vitamina C: kivi 71
 Unesite ime vocke i njenu kolicinu vitamina C: banana 8.7
 Unesite ime vocke i njenu kolicinu vitamina C: pomorandza 70.8
 Unesite ime vocke i njenu kolicinu vitamina C:
 Voce sa najvise vitamina C je: limun
```

Zadatak 2.9.4 Definisati strukturu `Grad` koja sadrži ime grada i njegovu prosečnu temperaturu u toku decembra. Napisati funkcije:

(a) `void ucitaj(Grad gradovi[], int n)` koja učitava sa standardnog ulaza podatke o n gradova.

(b) `void ispisi(Grad gradovi[], int n)` koja ispisuje podatke o gradovima koji imaju idealnu temperaturu za klizanje: od 3 do 8 stepeni celzijusa.

Napisati program koji učitava imena n gradova i njihove prosečne temperature, a zatim ispisuje imena gradova sa idealnom temperaturom za klizanje. Pretpostaviti da je maksimalan broj gradova 50 i da je maksimalna dužina imena grada 20 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj gradova: 4
 Unesite grad i temperaturu:
 Beograd 7
 Unesite grad i temperaturu:
 Uzice 1.5
 Unesite grad i temperaturu:
 Subotica 4
 Unesite grad i temperaturu:
 Zrenjanin 9
 Gradovi sa idealnom temperaturom
 za klizanje u decembru:
 Beograd
 Subotica
```

Primer 2

```
Interakcija sa programom:
 Unesite broj gradova: 2
 Unesite grad i temperaturu:
 Varsava 11
 Unesite grad i temperaturu:
 Prag 2
 Gradovi sa idealnom temperaturom
 za klizanje u decembru:
```

Zadatak 2.9.5 Definisati strukturu `ParReci` koja sadrži reč na srpskom jeziku i odgovarajući prevod na engleski jezik. Napisati program koji do kraja

ulaza učitava sve parove reči, a potom za rečenicu koja se zadaje u jednoj liniji ispisuje prevod. Ako je reč u rečenici nepoznata umesto nje ispisati nisku zvezdica čija dužina odgovara dužini nepoznate reči. Pretpostaviti da je maksimalna dužina reči 50 karaktera, maksimalan broj parova reči 100, a maksimalna dužina rečenice 100 karaktera.

Primer 1

```
Interakcija sa programom:
 Unesite reci i njihove prevode:
 zima winter
 godina year
 sreca happiness
 programiranje programming
 caj tea
 Unesite recenicu za prevod:
 piti caj zimi je sreca
 **** tea **** ** happiness
```

Primer 2

```
Interakcija sa programom:
 Unesite reci i njihove prevode:
 je is
 zima winter
 pas dog
 sreca happiness
 prijatelj friend
 solja cup
 covek man
 Unesite recenicu za prevod:
 pas je covekov najbolji prijatelj
 dog is ******* ******** friend
```

Zadatak 2.9.6 Statistički zavod Srbije istražuje kako rade obdaništa u Srbiji. Za svako obdanište poznat je spisak koji sadrži broj dece u grupi, a zatim i ocene koje je svako dete dalo o radu obdaništa. Definisati strukturu `Dete` koja sadrži polja pol (`m` ili `z`), broj godina (od 3 do 6) i ocenu koju je dete dalo radu obdaništa (od 1 do 5). Napisati program koji učitava broj dece u grupi, a zatim i informacije o svakom detetu. Ispisati, na tri decimale, prosečnu ocenu koje je obdanište dobilo od dece sa unetim polom i brojem godina. Pretpostaviti da je maksimalan broj dece u obdaništu 200. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj dece u grupi: 5
 Unesite podatke za svako dete (pol,
 broj godina i ocenu):
 m 3 5
 z 3 4
 m 4 2
 m 5 4
 m 3 4
 Unesite pol i broj godina za
 statistiku: m 3
 Prosecna ocena je: 4.500.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj dece u grupi: 10
 Unesite podatke za svako dete (pol,
 broj godina i ocenu):
 m 3 5
 z 4 4
 m 5 4
 z 4 3
 z 3 2
 z 4 5
 m 6 5
 z 4 4
 z 4 5
 m 6 3
 Unesite pol i broj godina za
 statistiku: z 4
 Prosecna ocena je: 4.200.
```

Primer 3

```
Interakcija sa programom:
Unesite broj dece u grupi: 15
Unesite podatke za svako dete (pol,
broj godina i ocenu):
m 3 2
z 7 5
Greska: neispravan broj godina.
```

Primer 4

```
Interakcija sa programom:
Unesite broj dece u grupi: 2
Unesite podatke za svako dete (pol,
broj godina i ocenu):
m 3 2
z 3 5
Unesite pol i broj godina za
statistiku: h 5
Greska: neispravan pol.
```

Zadatak 2.9.7 Definisati strukturu kojom se opisuje student. Student se opisuje svojim imenom i prezimenom, smerom (R, I, V, N, T, M) i prosečnom ocenom. Napisati program koji učitava podatke o n studenata, a zatim i informaciju o smeru i ispisuje imena i prezimena onih studenta koji su sa datog smera, kao i podatke studenta koji ima najveći prosek. Ako ima više takvih studenata ispisati podatke o svima. Pretpostaviti da je maksimalan broj studenata 2000, a maksimalna dužina imena i prezimena po 30 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
Unesite broj studenata: 5
Unesite podatke o studentima:
0. student: Kocic Marija R 9.14
1. student: Tanja Mratinkovic R 7.88
2. student: Mihailo Simic N 8.44
3. student: Milena Medar I 9.14
4. student: Ljubica Mihic N 9.00
Unesite smer: R
Studenti sa R smera:
Kocic Marija
Tanja Mratinkovic
---------------------
Svi studenti koji imaju maksimalni prosek:
Kocic Marija, R, 9.14
Milena Medar, I, 9.14
```

Primer 2

```
Interakcija sa programom:
Unesite broj studenata: 4
Unesite podatke o studentima:
0. student: Djordje Lazarevic N 9.05
1. student: Minja Peric W 7.70
Greska: neispravan unos smera.
```

Zadatak 2.9.8 Definisati strukturu `Djak` koja sadrži ime đaka i 9 ocena (ocene su celi brojevi od 1 do 5). Napisati program koji učitava podatke o đacima sve do kraja ulaza i na standardni izlaz ispisuje prvo imena nedovoljnih đaka, a zatim imena odličnih đaka. Đak je nedovoljan ako ima barem jednu jedinicu, a odličan ako ima prosek ocena veći ili jednak 4.5. Pretpostaviti da je maksimalna dužina imena đaka 20 karaktera, kao i da je maksimalan broj đaka 30. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite podatke o djaku:
 Maja 4 5 2 3 4 4 3 3 4
 Unesite podatke o djaku:
 Nikola 5 4 5 5 5 4 4 5 5
 Unesite podatke o djaku:
 Jasmina 2 2 1 1 2 3 3 1 3
 Unesite podatke o djaku:
 Pera 5 4 5 3 5 5 1 5 5
 Unesite podatke o djaku:
 Pavle 4 3 2 4 3 2 4 3 2
 Unesite podatke o djaku:

 NEDOVOLJNI: Jasmina Pera
 ODLICNI: Nikola
```

Primer 2

```
Interakcija sa programom:
 Unesite podatke o djaku:
 Uros 3 4 2 3 4 2 3 4 4
 Unesite podatke o djaku:
 Nebojsa 4 5 5 5 4 5 5 5 5
 Unesite podatke o djaku:
 Sreten 2 3 2 4 5 4 4 4 2
 Unesite podatke o djaku:

 NEDOVOLJNI:
 ODLICNI: Nebojsa
```

Primer 3

```
Interakcija sa programom:
 Unesite podatke o djaku:
 Mirko 2 3 4 4 4 3 3 3 4
 Unesite podatke o djaku:
 Mihailo 2 3 10 5 5 2 3 4 2
 Greska: neispravna ocena.
```

Zadatak 2.9.9 Defnisati strukturu `Osoba` kojom se opisuje jedan unos u imenik. Za svaku osobu su dati podaci: ime, prezime i imejl adresa. Napisati program koji učitava ceo broj n, a zatim podatke o n osoba. Ispisati imena i prezimena svih osoba koje imaju imejl adresu koja se završava sa `@gmail.com`. Pretpostaviti da je maksimalan broj osoba 50, kao i da je maksimalna dužina imena osobe 20 karaktera, prezimena 30 karaktera, a imejl adrese 50 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci. Napomena: *Može se smatrati da je svaka imejl adresa dobro zadata i sadrži samo jedno pojavljivanje znaka `@`.*

Primer 1

```
Interakcija sa programom:
 Unesite broj osoba: 3
 Unesite podatke o osobama
 (ime, prezime i imejl adresu):
 Dusko Dugousko dusko@yahoo.com
 Pink Panter panter@gmail.com
 Pera Detlic pd@gmail.com
 Vlasnici gmail naloga su:
 Pink Panter
 Pera Detlic
```

Primer 2

```
Interakcija sa programom:
 Unesite broj osoba: 3
 Unesite podatke o osobama
 (ime, prezime i imejl adresu):
 Homer Simpson homer@yahoo.com
 Mardz Simpson mardz@matf.bg.ac.rs
 Nema vlasnika gmail naloga.
```

*** Zadatak 2.9.10** Napisati program koji izračunava prosečnu cenu jedne potrošačke korpe. Potrošačka korpa se sastoji od broja kupljenih artikala i niza kupljenih artikala. Svaki artikal određen je svojim nazivom, količinom i cenom. Program treba da učita broj potrošača n, zatim podatke za n potrošačkih korpi i da na osnovu učitanih podataka izračuna prosečnu cenu potrošačke korpe. Program ispisuje na dve decimale račune svake potrošačke korpe i na kraju ispisuje prosečnu cenu potrošačke korpe. Pretpostaviti da je maksimalan broj potrošačkih

korpi 100, maksimalan broj artikala u korpi 20 i da naziv svakog artikla sadrži maksimalno 30 karaktera. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj potrosackih korpi: 3
 Unesite podatke o korpi:
 Broj artikala: 4
 Unesite artikal (naziv, kolicinu i cenu): jabuke 10 22.4
 Unesite artikal (naziv, kolicinu i cenu): dezodorans 1 120.99
 Unesite artikal (naziv, kolicinu i cenu): C_supa 3 36.56
 Unesite artikal (naziv, kolicinu i cenu): sunka 1 230.99
 Unesite podatke o korpi:
 Broj artikala: 2
 Unesite artikal (naziv, kolicinu i cenu): Jafa_keks 1 55.78
 Unesite artikal (naziv, kolicinu i cenu): Najlepse_zelje 1 62.99
 Unesite podatke o korpi:
 Broj artikala: 3
 Unesite artikal (naziv, kolicinu i cenu): prasak_za_ves 1 1199.99
 Unesite artikal (naziv, kolicinu i cenu): omeksivac 1 279.99
 Unesite artikal (naziv, kolicinu i cenu): protiv_kamenca 1 699.99

 Korpa 0:
         jabuke 10 22.40
         dezodorans 1 120.99
         C_supa 3 36.56
         sunka 1 230.99
 ------------------------
         ukupno: 685.66

 Korpa 1:
         Jafa_keks 1 55.78
         Najlepse_zelje 1 62.99
 ------------------------
         ukupno: 118.77

 Korpa 2:
         prasak_za_ves 1 1199.99
         omeksivac 1 279.99
         protiv_kamenca 1 699.99
 ------------------------
         ukupno: 2179.97

 Prosecna cena potrosacke korpe: 994.80
```

Zadatak 2.9.11 Definisati strukturu `Lopta` sa poljima `poluprecnik` (ceo broj u centimetrima) i `boja` (enumeracioni tip koji uključuje plavu, žutu, crvenu i zelenu boju). Napisati funkcije:

(a) `void ucitaj(Lopta niz[], int n)` koja učitava podatke o n lopti u niz.

(b) `double ukupna_zapremina(Lopta niz[], int n)` koja računa ukupnu zapreminu svih lopti.

(c) `int broj_crvenih(Lopta niz[], int n)` koja prebrojava koliko ima crvenih lopti u nizu.

Napisati program koji učitava informacije o n lopti i ispisuje ukupnu zapreminu i broj crvenih lopti. Pretpostaviti da je maksimalan broj lopti 50. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj lopti: 4
 Unesite poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1.lopta: 4 1
 2.lopta: 1 3
 3.lopta: 2 3
 4.lopta: 10 4
 Ukupna zapremina: 4494.57
 Broj crvenih lopti: 2
```

Primer 2

```
Interakcija sa programom:
 Unesite broj lopti: 8
 Unesite poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1. lopta: 1 2
 2. lopta: 2 10
 Greska: neispravan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite broj lopti: 8
 Unesite poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1. lopta: 2 1
 2. lopta: 30 3
 3. lopta: 7 3
 4. lopta: 4 1
 5. lopta: 5 2
 6. lopta: 6 2
 7. lopta: 12 3
 8. lopta: 14 2
 Ukupna zapremina: 134996.34
 Ukupno crvenih lopti: 3
```

Zadatak 2.9.12 Napisati program za predstavljanje poligona i izračunavanje dužine njegovih stranica i obima.

(a) Definisati strukturu `Tacka` kojom se opisuje tačka Dekartovske ravni čije su x i y koordinate podaci tipa `double`.

(b) Definisati funkciju `double rastojanje(const Tacka *A, const Tacka *B)` koja izračunava rastojanje između dve tačke.

(c) Definisati funkciju `int ucitaj_poligon(Tacka poligon[], int n)` koja učitava maksimalno n puta po dve vrednosti tipa `double` (koje predstavljaju koordinate temena poligona) i upisuje ih u zadati niz tačaka. Funkcija vraća broj uspešno učitanih tačaka.

(d) Definisati funkciju `double obim_poligona(Tacka poligon[], int n)` koja izračunava obim poligona sa n temena u zadatom nizu. UPUTSTVO: *Prilikom računanja obima ne zaboraviti stranicu koja spaja poslednje i prvo teme.*

(e) Definisati funkciju `double maksimalna_stranica(Tacka poligon[], int n)` koja izračunava dužinu najduže stranice poligona sa n temena u zadatom nizu.

(f) Napisati funkciju `double povrsina_trougla(const Tacka *A, const Tacka *B, const Tacka *C)` koja izračunava površinu trougla čija su temena `A`, `B` i `C`.

(g) Napisati funkciju `double povrsina_poligona(Tacka poligon[], int n)` koja izračunava površinu konveksnog poligona. UPUTSTVO: *Zadatak se može rešiti podelom poligona na trouglove i korišćenjem funkcije* ***povrsina_trougla****.*

Napisati program koji učitava poligon sa maksimalno n temena i za učitani poligon ispisuje na tri decimale obim, dužinu najduže stranice i površinu. Pretpostaviti da je uneti poligon konveksan. Poligon mora imati barem tri temena. Pretpostaviti da je maksimalan broj temena 1000. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite maksimalan broj temena poligona: 10
 Unesite temena poligona:
 0 0
 0 6
 3 3
 Obim poligona je 14.485.
 Duzina maksimalne stranice je 6.000.
 Povrsina poligona je 9.000.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite maksimalan broj temena poligona: 10
 Unesite temena poligona:
 0 0
 12 0
 13 2
 16 5
 20 10
 18 15
 15 20
 10 20
 8 15
 3 4
 Obim poligona je 63.566.
 Duzina maksimalne stranice je 12.083.
 Povrsina poligona je 247.500.
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite maksimalan broj temena poligona: 4
 0 0
 Greska: poligon mora imati bar tri tacke.
```

*** Zadatak 2.9.13** Definisati strukturu `Izraz` kojom se opisuje numerički izraz nad celim brojevima koji se sastoji od dva celobrojna operanda i numeričke operacije (sabiranje, oduzimanje, množenje ili celobrojno deljenje).

(a) Napisati funkciju `int korektan_izraz(const Izraz *izraz)` koja ispituje da li je dati izraz korektno zadat i vraća jedinicu ako jeste, a nulu inače.

Podrazumeva se da je izraz korektno zadat ako je operacija $+$, $-$, $*$ ili / i u slučaju deljenja drugi operand je različit od 0.

(b) Napisati funkciju `int vrednost(const Izraz *izraz)` koja za dati izraz određuje vrednost izraza.

(c) Napisati funkciju `void ucitaj(Izraz izrazi[], int n)` koja učitava izraze. Funkcija treba da učita sa standardnog ulaza n izraza koji su zadati prefiksno — prvo operacija, a potom dva operanda.

Napisati program koji učitava prirodan broj n, a zatim n izraza u prefiksnoj notaciji. Program treba da ispiše maksimalnu vrednost unetih izraza i sve izraze čija vrednost je manja od polovine maksimalne vrednosti. Pretpostaviti da je maksimalan broj izraza 100. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite broj izraza: 4
 Unesite izraze u prefiksnoj notaciji:
 + 10 4
 - 9 2
 * 11 2
 / 7 3
 Maksimalna vrednost izraza: 22
 Izrazi cija je vrednost manja
 od polovine maksimalne vrednosti:
 9 - 2 = 7
 7 / 3 = 2
```

Primer 2

```
Interakcija sa programom:
 Unesite broj izraza: 10
 Unesite izraze u prefiksnoj notaciji:
 + 10 2
 - -678 34
 * 77 2
 + 1000 -23
 + 102 4
 - 200 23
 / 67 12
 / 1000 2
 * 44 6
 / 13 1
 Maksimalna vrednost izraza: 977
 Izrazi cija je vrednost manja
 od polovine maksimalne vrednosti:
 10 + 2 = 12
 -678 - 34 = -712
 77 * 2 = 154
 102 + 4 = 106
 200 - 23 = 177
 67 / 12 = 5
 44 * 6 = 264
 13 / 1 = 13
```

Primer 3

```
Interakcija sa programom:
 Unesite broj izraza: 3
 Unesite izraze u prefiksnoj notaciji:
 * 1 2
 / 3 0
 Greska: deljenje nulom.
```

*** Zadatak 2.9.14** Definisati strukturu kojom se opisuje polinom. Polinom je dat svojim stepenom i realnim koeficijentima.

(a) Napisati funkciju `int ucitaj(Polinom niz[])` koja sa standardnog ulaza učitava polinome sve do kraja ulaza. Polinomi su zadati stepenom i koeficijentima počevši od slobodnog člana. Funkcija kao povratnu vrednost vraća broj učitanih polinoma.

(b) Napisati funkciju `void ispis(const Polinom *p)` koja ispisuje polinom stepena n sa koeficijentima k_0, k_1, ..., k_n u obliku $k_0 \pm k_1 * x \pm k_2 * x\hat{}2 \pm k_3 * x\hat{}3 \pm \ldots \pm k_n * x\hat{}n$. Na mesto znaka $\pm$ zapisati odgovarajući znak, $+$ ili $-$, u zavisnosti od znaka odgovarajućeg koeficijenta. Koeficijente ispisivati na dve decimale. Koeficijente koji su jednaki 0 ne ispisivati.

(c) Napisati funkciju `void integral(const Polinom *p, Polinom *tekuci_integral)` koja za dati polinom `p` određuje njegov integral `tekuci_integral`. Za vrednost slobodnog člana integrala uzeti vrednost 0.

Napisati program koji učitava polinome do kraja ulaza i za svaki učitani polinom određuje i ispisuje njegov integral. Pretpostaviti da je maksimalan broj polinoma 100, a maksimalan stepen polinoma 10. U slučaju neispravnog unosa, ispisati odgovarajuću poruku o grešci.

Primer 1

```
Interakcija sa programom:
 Unesite stepen: 3
 Unesite koeficijente polinoma:
 1 0 3 1
 Unesite stepen: 4
 Unesite koeficijente polinoma:
 7 9 4 0 4
 Unesite stepen:

 Integrali su:
 1.00*x + 1.00*x^3 + 0.25*x^4
 7.00*x + 4.50*x^2 + 1.33*x^3 + 0.80*x^5
```

Primer 2

```
Interakcija sa programom:
 Unesite stepen: 3
 Unesite koeficijente polinoma:
 1 0 -4 1
 Unesite stepen: 2
 Unesite koeficijente polinoma:
 1 2 -3
 Unesite stepen: 1
 Unesite koeficijente polinoma:
 0 -1
 Unesite stepen:

 Integrali su:
 1.00*x - 1.33*x^3 + 0.25*x^4
 1.00*x + 1.00*x^2 - 1.00*x^3
 -0.50*x^2
```

2.10 Rešenja

Rešenje 2.9.1

```
#include <stdio.h>
#include <math.h>

/* Struktura koja opisuje kompleksni broj. */
typedef struct {
  float re;
  float im;
} KompleksanBroj;

/* Kada se neka promenljiva zadaje kao argument funkcije, obicno se
   prenosi po vrednosti (bez pokazivaca), ako se ona nece menjati u
   funkciji ili po adresi (preko pokazivaca), ako ce se njena
   vrednost promeniti u funkciji.

   Prilikom poziva funkcije, za svaki argument funkcije kreira se
   promenljiva koja predstavlja lokalnu kopiju argumenta i koja
   prestaje da postoji po zavrsetku funkcije. S obzirom da se
   strukuture sastoje od vise polja, zauzimaju vise memorije nego
   nestrukturne promenljive. Zbog toga je za njihovo kopiranje
   potrebno vise vremena i vise memorijskih resursa nego za
   kopiranje nestrukturnih promenljivih.

   Da bi program bio efikasniji, korisno je da se struktura uvek
   prenosi po adresi (preko pokazivaca), bez obzira da li ce se
   ona u toj funkciji menjati ili ne. Pokazivac na strukturu
   zauzima manje memorije nego sama struktura pa je izrada njegove
   kopije brza, a kopija pokazivaca uzima manji memorijski prostor
   nego kopija strukture.

   Kada se strukturna promenljiva prenosi u funkciju po adresi
   (preko pokazivaca), tada postoji mogucnost da se njena polja
   menjaju u funkciji. Ukoliko to nije potrebno, uz argument se
   dodaje kljucna rec const. Na taj nacin, u slucaju pokusaja
   izmene strukturne promenljive koja je prosledjena kao const,
   kompajler ce prijaviti gresku. Na ovaj nacin se obezbedjuje da
   promenljiva koja je preneta po adresi ne bude cak ni slucajno
   izmenjena u funkciji. */

   /* Funkcija izracunava zbir kompleksnih brojeva. */
KompleksanBroj saberi(const KompleksanBroj *a,
                      const KompleksanBroj *b) {
  KompleksanBroj c;
  c.re = a->re + b->re;
  c.im = a->im + b->im;
  return c;
}

```

```
/* Funkcija izracunava razliku kompleksnih brojeva. */
KompleksanBroj oduzmi(const KompleksanBroj *a,
                      const KompleksanBroj *b) {
  KompleksanBroj c;
  c.re = a->re - b->re;
  c.im = a->im - b->im;
  return c;
}

/* Funkcija izracunava proizvod kompleksnih brojeva. */
KompleksanBroj pomnozi(const KompleksanBroj *a,
                       const KompleksanBroj *b) {
  KompleksanBroj c;
  c.re = a->re * b->re - a->im * b->im;
  c.im = b->re * a->im + a->re * b->im;
  return c;
}

/* Funkcija izracunava kolicnik kompleksnih brojeva. */
KompleksanBroj podeli(const KompleksanBroj *a,
                      const KompleksanBroj *b,
                      int *postoji_kolicnik) {
  KompleksanBroj c;

  if (b->re != 0 || b->im != 0) {
    c.re = (a->re * b->re + a->im * b->im) /
        (b->re * b->re + b->im * b->im);
    c.im = (b->re * a->im - a->re * b->im) /
        (b->re * b->re + b->im * b->im);
  } else {
    printf("Kolicnik ne postoji.\n");
    *postoji_kolicnik = 0;
  }

  return c;
}

/* Funkcija ispisuje kompleksan broj. */
void ispisi(const KompleksanBroj *c){
    /* Ukoliko je imaginarni deo negativan, njegov zapis vec
       ukljucuje znak, pa se zato uzima njegova apsolutna
       vrednost. */
    printf("%.2f%c%.2f*i\n", c->re, c->im > 0 ? '+' : '-',
           fabs(c->im));
}

int main() {
  /* Deklaracije potrebnih promenlivih. */
  KompleksanBroj a, b, c;
  int postoji_kolicnik = 1;

  /* Ucitavanje kompleksnih brojeva. */
```

```
  printf("Unesite realni i imaginarni deo prvog broja: ");
  scanf("%f%f", &a.re, &a.im);
  printf("Unesite realni i imaginarni deo drugog broja: ");
  scanf("%f%f", &b.re, &b.im);

  /* Ispis zbira. */
  c = saberi(&a, &b);
  printf("Zbir: ");
  ispisi(&c);

  /* Ispis razlike. */
  c = oduzmi(&a, &b);
  printf("Razlika: ");
  ispisi(&c);

  /* Ispis proizvoda. */
  c = pomnozi(&a, &b);
  printf("Proizvod: ");
  ispisi(&c);

  /* Ispis kolicnika. */
  c = podeli(&a, &b, &postoji_kolicnik);
  if (postoji_kolicnik) {
    printf("Kolicnik: ");
    ispisi(&c);
  }

  return 0;
}
```

Rešenje 2.9.2

```
#include <stdio.h>
#include <stdlib.h>

/* Struktura koja opisuje razlomak. */
typedef struct {
  int brojilac;
  int imenilac;
} Razlomak;

/* Funkcija Euklidovim algoritmom racuna najveci zajednicki delilac
   brojeva a i b. */
int nzd(int a, int b) {
  int ostatak;

  while (b != 0) {
    ostatak = a % b;
    a = b;
    b = ostatak;
  }
```

```
21    return a;
  }
23
  /* Funkcija vraca razlomak koji se dobija deljenjem imenioca i
25     brojioca njihovim najvecim zajednickim deliocem. */
  void skrati(Razlomak *r) {
27    int nzd_razlomka = nzd(r->brojilac, r->imenilac);
    r->brojilac /= nzd_razlomka;
29    r->imenilac /= nzd_razlomka;
  }
31
  /* Funkcija racuna zbir razlomaka a i b. */
33 Razlomak saberi(const Razlomak *a, const Razlomak *b) {
    Razlomak c;
35
    c.brojilac = a->brojilac * b->imenilac +
37                 b->brojilac * a->imenilac;
    c.imenilac = a->imenilac * b->imenilac;
39    skrati(&c);

41    return c;
  }
43
  /* Funkcija racuna proizvod razlomaka a i b. */
45 Razlomak pomnozi(const Razlomak *a, const Razlomak *b) {
    Razlomak c;
47
    c.brojilac = a->brojilac * b->brojilac;
49    c.imenilac = a->imenilac * b->imenilac;
    skrati(&c);
51
    return c;
53 }

55 int main() {
    /* Deklaracije potrebnih promenljivih. */
57    int n, i;
    Razlomak suma, proizvod, r;
59
    /* Ucitavanje broja razlomaka i provera ispravnosti ulaza. */
61    printf("Unesite broj razlomaka: ");
    scanf("%d", &n);
63    if (n <= 0) {
      printf("Greska: neispravan unos.\n");
65      exit(EXIT_FAILURE);
    }
67
    /* Inicijalizacija sume i proizvoda. */
69    suma.brojilac = 0;
    suma.imenilac = 1;
71    proizvod.brojilac = 1;
```

```
  proizvod.imenilac = 1;

  /* Ucitavanje razlomaka i racunanje rezultata. */
  printf("Unesite razlomke:\n");
  for (i = 0; i < n; i++) {
    scanf("%d%d", &r.brojilac, &r.imenilac);

    if (r.imenilac == 0) {
      printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
    }

    suma = saberi(&suma, &r);
    proizvod = pomnozi(&proizvod, &r);
  }

  /* Ispis rezultata. */
  printf("Suma svih razlomaka: %d/%d\n", suma.brojilac,
         suma.imenilac);
  printf("Proizvod svih razlomaka: %d/%d\n", proizvod.brojilac,
         proizvod.imenilac);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.3

```
#include <stdio.h>
#include <string.h>

#define MAKS_IME 21
#define MAKS_VOCKI 50

/* Struktura koja opisuje vocku. */
typedef struct {
  char ime[MAKS_IME];
  float vitamin;
} Vocka;

/* Funkcija ucitava podatke o vockama u niz struktura. Kao
   povratnu vrednost vraca broj ucitanih vocki. */
int ucitaj(Vocka niz[]) {
  int i = 0;

  /* Ucitavanje vocki do kraja ulaza ili do popunjavanja niza. */
  do {
    printf("Unesite ime vocke i njenu kolicinu vitamina C: ");
    if(scanf("%s%f", niz[i].ime, &niz[i].vitamin) == EOF)
      break;

    i++;
```

```
  } while (i < MAKS_VOCKI);

  return i;
}

/* Funkcija pronalazi vocku sa najvise vitamina C. */
Vocka vocka_sa_najvise_vitamina(Vocka niz[], int n) {
  /* Pronalazak pozicije vocke sa najvise vitamina C. */
  int maks_i = 0, i;
  for (i = 1; i < n; i++)
    if (niz[i].vitamin > niz[maks_i].vitamin)
      maks_i = i;

  /* Kao povratna vrednost se vraca vocka na poziciji maks_i. */
  return niz[maks_i];
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  Vocka vocke[MAKS_VOCKI], najzdravija;
  int n;

  /* Ucitavanje ulaza. */
  n = ucitaj(vocke);

  /* Ispis rezultata. */
  najzdravija = vocka_sa_najvise_vitamina(vocke, n);
  printf("Voce sa najvise vitamina C je: %s\n", najzdravija.ime);

  return 0;
}
```

Rešenje 2.9.4 Pogledajte zadatak 2.9.3.

Rešenje 2.9.5

```
#include <stdio.h>
#include <string.h>

#define MAKS_REC 21
#define MAKS_BROJ_RECI 100

/* Struktura koja opisuje par reci. */
typedef struct {
  char sr[MAKS_REC];
  char en[MAKS_REC];
} ParReci;

/* Funkcija ucitava parove reci u recnik. */
int ucitaj(ParReci recnik[]) {
  int i = 0;
```

```
16   char sr[MAKS_REC], en[MAKS_REC];

18   /* Ucitavanje parovi reci sa standardnog ulaza sve do kraja
        ulaza. */
20   printf("Unesite reci i njihove prevode:\n");
     while (scanf("%s %s", sr, en) != EOF) {
22     if (i == MAKS_BROJ_RECI)
         break;
24
       strcpy(recnik[i].sr, sr);
26     strcpy(recnik[i].en, en);

28     i++;
     }
30
     return i;
32 }

34 /*
      Funkcija u recniku koji sadrzi n reci trazi prevod reci rec i
36    upisuje ga u prevod. Ukoliko se rec ne nalazi u recniku, prevod
      se sastoji od zvezdica pri cemu broj zvezdica odgovara duzini
38    nepoznate reci. */
   void pronadji_prevod(ParReci recnik[], int n, char rec[],
40                       char prevod[]) {
     int i;
42
     /* Pretraga reci. */
44   for (i = 0; i < n; i++) {
       if (strcmp(recnik[i].sr, rec) == 0) {
46       strcpy(prevod, recnik[i].en);
         return;
48     }
     }
50
     /* Ukoliko rec nije pronadjena, formira se prevod reci koji se
52      sastoji od zvezdica. */
     for (i = 0; rec[i]; i++)
54     prevod[i] = '*';
     prevod[i] = '\0';
56 }

58 int main() {
     /* Deklaracije potrebnih promenljivih. */
60   ParReci recnik[MAKS_BROJ_RECI];
     int n;
62   char rec[MAKS_REC], prevod[MAKS_REC];
     char c;
64
     /* Ucitavanje parova reci u recnik. */
66   n = ucitaj(recnik);
```

```
68    /* Ucitavanje recenice i ispis njenog prevoda. */
      printf("Unesite recenicu za prevod: \n");
70    do {
        /* Ucitava se rec po rec date recenice i pronalazi se njen
72         prevod. */
        scanf("%s", rec);
74      pronadji_prevod(recnik, n, rec, prevod);
        printf("%s ", prevod);
76
        /* Ukoliko je karakter iza reci znak za novi red, onda se
78         prekida sa unosom, a ako nije ucitava se sledeca rec. */
        c = getchar();
80    } while (c != '\n');

82    putchar('\n');

84    return 0;
}
```

Rešenje 2.9.6

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   #define MAKS_DECE 200
5
   /* Struktura koja opisuje dete. */
7  typedef struct {
     char pol;
9    int broj_godina;
     int ocena;
11 } Dete;

13 /* Funkcija ucitava podatke o deci i proverava ispravnost unetih
      podataka. */
15 void ucitaj(Dete niz[], int n) {
     char blanko;
17   int i;
     printf("Unesite podatke za svako dete (pol, broj godina i "
19          "ocenu):\n");
     for (i = 0; i < n; i++) {
21     scanf("%c%c%d%d", &blanko, &niz[i].pol, &niz[i].broj_godina,
            &niz[i].ocena);
23
       /* Provera ispravnosti unosa. */
25     if (niz[i].pol != 'm' && niz[i].pol != 'z') {
         printf("Greska: neispravan pol.\n");
27       exit(EXIT_FAILURE);
       }
29     if (niz[i].broj_godina > 6 || niz[i].broj_godina < 3) {
         printf("Greska: neispravan broj godina.\n");
```

```
31        exit(EXIT_FAILURE);
      }
33      if (niz[i].ocena < 1 || niz[i].ocena > 5) {
        printf("Greska: neispravna ocena.\n");
35        exit(EXIT_FAILURE);
      }
37    }
  }
39
  int main() {
41    /* Deklaracija potrebnih promenljivih. */
    int n, i, broj_godina;
43    Dete niz[MAKS_DECE];
    char blanko, pol;
45    int suma, broj_dece;

47    /* Ucitavanje broja dece i provera ispravnosti ulaza. */
    printf("Unesite broj dece u grupi: ");
49    scanf("%d", &n);
    if (n <= 0 || n > MAKS_DECE) {
51      printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
53    }

55    /* Ucitavanje podataka o deci. */
    ucitaj(niz, n);
57
    /* Ucitavanje trazenih podataka. */
59    printf("Unesite pol i broj godina za statistiku: ");
    scanf("%c%c%d", &blanko, &pol, &broj_godina);
61
    /* Provera ispravnosti unetih podataka. */
63    if (pol != 'm' && pol != 'z') {
      printf("Greska: neispravan pol.\n");
65      exit(EXIT_FAILURE);
    }
67    if (broj_godina > 6 || broj_godina < 3) {
      printf("Greska: neispravan broj godina.\n");
69      exit(EXIT_FAILURE);
    }
71
    /* Racunanje prosecne ocene dece ciji se pol i broj godina
73       poklapaju sa unetim. */
    suma = 0;
75    broj_dece = 0;
    for (i = 0; i < n; i++)
77      if (niz[i].pol == pol && niz[i].broj_godina == broj_godina) {
        suma += niz[i].ocena;
79        broj_dece++;
      }
81
    /* Ispis rezultata. */
```

```
  if (broj_dece == 0)
    printf("Ne postoje deca sa takvim karakteristikama.\n");
  else
    printf("Prosecna ocena je: %.3lf.\n",
           (double) suma / broj_dece);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.7

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_STUDENATA 2000
#define MAKS_NISKA 31

/* Struktura koja opisuje studenta. */
typedef struct Student {
  char ime[MAKS_NISKA];
  char prezime[MAKS_NISKA];
  char smer;
  float prosek;
} Student;

/* Funkcija ucitava podatke o studentima u niz. */
void ucitaj(Student niz[], int n) {
  int i;

  printf("Unesite podatke o studentima:\n");
  for (i = 0; i < n; i++) {
    printf("%d. student: ", i);
    scanf("%s %s %c %f", niz[i].ime, niz[i].prezime,
          &niz[i].smer, &niz[i].prosek);

    if (niz[i].smer != 'R' && niz[i].smer != 'I' &&
        niz[i].smer != 'V' && niz[i].smer != 'N' &&
        niz[i].smer != 'T' && niz[i].smer != 'O') {
      printf("Greska: neispravan unos smera.\n");
      exit(EXIT_FAILURE);
    }
  }
}

/* Funkcija ispisuje podatke o studentu. */
void ispisi(const Student *s) {
  printf("%s %s, %c, %.2f\n", s->ime, s->prezime, s->smer,
         s->prosek);
}

/* Funkcija racuna najveci prosek. */
```

```
float najveci_prosek(Student studenti[], int n) {
  float maks_prosek;
  int i;

  maks_prosek = studenti[0].prosek;
  for (i = 1; i < n; i++)
    if (maks_prosek < studenti[i].prosek)
      maks_prosek = studenti[i].prosek;

  return maks_prosek;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  Student studenti[MAKS_STUDENATA];
  int n, i;
  float maks_prosek;
  char smer;

  /* Ucitavanje broja studenata i provera ispravnosti ulaza. */
  printf("Unesite broj studenata: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS_STUDENATA) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje podataka o studentima. */
  ucitaj(studenti, n);

  /* Ucitavanje smera. Pre smera se preskace novi red koji je unet
     nakon podataka o poslednjem studentu. */
  printf("Unesite smer: ");
  getchar();
  scanf("%c", &smer);
  if (smer != 'R' && smer != 'I' && smer != 'V' && smer != 'N' &&
      smer != 'T' && smer != 'O') {
    printf("Greska: neispravan unos smera.\n");
    exit(EXIT_FAILURE);
  }

  /* Ispis studenata sa unetog smera. */
  printf("Studenti sa %c smera:\n", smer);
  for (i = 0; i < n; i++)
    if (studenti[i].smer == smer)
      printf("%s %s\n", studenti[i].ime, studenti[i].prezime);
  printf("----------------------\n");

  /* Racunanje najveceg proseka. */
  maks_prosek = najveci_prosek(studenti, n);

  /* Ispis svih studenata sa najvecim prosekom. */
```

```
  printf("Svi studenti koji imaju maksimalni prosek:\n");
  for (i = 0; i < n; i++)
    if (studenti[i].prosek == maks_prosek)
      ispisi(&studenti[i]);

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.8

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_IME 21
#define BROJ_OCENA 9
#define MAKS_DJAKA 30

/* Struktura koja opisuje djaka. */
typedef struct {
  char ime[MAKS_IME];
  int ocena[BROJ_OCENA];
} Djak;

/* Funkcija proverava ispravnost date ocene. */
void provera_ocene(int ocena) {
  if (ocena < 1 || ocena > 5) {
    printf("Greska: neispravna ocena.\n");
    exit(EXIT_FAILURE);
  }
}

/* Funkcija ucitava podatke o djacima u niz. */
int ucitaj(Djak niz[]) {
  int i = 0, j;

  while (i < MAKS_DJAKA) {
    printf("Unesite podatke o djaku: ");
    /* Ucitavanje imena. */
    if (scanf("%s", niz[i].ime) == EOF)
      break;

    /* Ucitavanje ocena. */
    for (j = 0; j < BROJ_OCENA; j++) {
      scanf("%d", &niz[i].ocena[j]);
      provera_ocene(niz[i].ocena[j]);
    }
    i++;
  }

  return i;
}
```

```
42
/* Funkcija racuna prosecnu ocenu datog djaka. */
44 float prosecna_ocena(const Djak *djak) {
  int j;
46  float suma = 0;
  for (j = 0; j < BROJ_OCENA; j++)
48    suma += djak->ocena[j];

50  return suma / BROJ_OCENA;
}
52
int main() {
54  /* Deklaracija potrebnih promenljivih. */
  Djak niz[MAKS_DJAKA];
56  int i, j, n;
  float prosek;
58
  /* Ucitavanje podataka o djacima. */
60  n = ucitaj(niz);

62  /* Ispis imena nedovoljnih djaka. */
  printf("\n\nNEDOVOLJNI: ");
64  for (i = 0; i < n; i++)
    for (j = 0; j < BROJ_OCENA; j++)
66      if (niz[i].ocena[j] == 1) {
        printf("%s ", niz[i].ime);
68      break;
      }
70  printf("\n");

72  /* Ispis imena odlicnih djaka. */
  printf("ODLICNI: ");
74  for (i = 0; i < n; i++) {
    prosek = prosecna_ocena(&niz[i]);
76    if (prosek >= 4.5)
      printf("%s ", niz[i].ime);
78  }
  printf("\n");
80
  exit(EXIT_SUCCESS);
82 }
```

Rešenje 2.9.9

```
1 #include <stdio.h>
  #include <string.h>
3 #include <stdlib.h>

5 #define MAKS_IME 21
  #define MAKS_PREZIME 31
7 #define MAKS_EMAIL 51
```

```
#define MAKS_OSOBA 50

/* Struktura koja opisuje osobu. */
typedef struct {
  char ime[MAKS_IME];
  char prezime[MAKS_PREZIME];
  char email[MAKS_EMAIL];
} Osoba;

/* I nacin: Funkcija proverava da li se prosledjeni email zavrsava
   sa "gmail.com" koriscenjem funkcije strtok. */
int gmail(char email[]) {
  /* Funkcija strtok "deli" nisku u podniske tako sto ih razdvaja
     na mestu na kom se nalazi prosledjeni delimiter (u ovom
     slucaju je to "@"). Na primer, ukoliko je
     email="pera.peric@gmail.com", funkcija deli ovu nisku na
     "pera.peric" i "gmail.com". */
  char *deo = strtok(email, "@");

  /* Kada se funkcija sledeci put pozove i pri tom pozivu se kao
     prvi argument navede NULL, tada funkcija vraca sledeci token u
     nizu, a to je u ovom slucaju "gmail.com". */
  deo = strtok(NULL, "");

  /* Ako se email zavrsava na "gmail.com", funkcija vraca 1, a u
     suprotnom 0. */
  return strcmp(deo, "gmail.com") == 0;
}

/* II nacin:
int gmail2(char email[])
{
  //Pronalazi se pokazivac na znak @.
  char* desni_deo = strchr(email, '@');

  //Poredi se niska koja pocinje jedan karakter posle @ sa
  //niskom "gmail.com".
  return strcmp(desni_deo+1, "gmail.com") == 0;
} */


int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n, i, postoji_gmail_adresa = 0;
  Osoba osobe[MAKS_OSOBA];

  /* Ucitavanje broja osoba i provera ispravnosti ulaza. */
  printf("Unesite broj osoba: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS_OSOBA) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
```

```
  }

  /* Ucitavanje podataka o osobama. */
  printf("Unesite podatke o osobama (ime, prezime i imejl adresu):\n"
    );
  for (i = 0; i < n; i++)
    scanf("%s%s%s", osobe[i].ime, osobe[i].prezime, osobe[i].email);

  /* Ispis rezultata. */
  for (i = 0; i < n; i++) {
    if (gmail(osobe[i].email)) {
      if (!postoji_gmail_adresa) {
        /* U ovu granu ce se uci samo kada se naidje na prvog
           vlasnika gmail naloga. */
        printf("Vlasnici gmail naloga su:\n");
        postoji_gmail_adresa = 1;
      }
      printf("%s %s\n", osobe[i].ime, osobe[i].prezime);
    }
  }

  /* Ukoliko se nije naislo ni na jednog vlasnika gmail naloga,
     promenljiva postoji_gmail_adresa ce ostati 0 i u tom slucaju
     se ispsuje odgovarajuca poruka. */
  if (!postoji_gmail_adresa)
    printf("Nema vlasnika gmail naloga.\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.10

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_ARTIKALA 20
#define MAKS_KORPI 100
#define MAKS_NAZIV 31

/* Struktura koja opisuje artikal. */
typedef struct {
  char naziv[MAKS_NAZIV];
  int kolicina;
  float cena;
} Artikal;

/* Struktura koja opisuje korpu. */
typedef struct {
  int broj_artikala;
  Artikal artikli[MAKS_ARTIKALA];
} Korpa;
```

```
20
   /* Funkcija ucitava jedan artikal i proverava ispravnost ucitanih
22    podataka. */
   void ucitaj_artikal(Artikal *a) {
24   printf("Unesite artikal (naziv, kolicinu i cenu): ");
     scanf("%s%d%f", a->naziv, &a->kolicina, &a->cena);
26
     if (a->kolicina <= 0) {
28     printf("Greska: neispravan unos kolicine (%d).\n", a->kolicina);
       exit(EXIT_FAILURE);
30   }

32   if (a->cena < 0) {
       printf("Greska: neispravan unos cene (%f).\n", a->cena);
34     exit(EXIT_FAILURE);
     }
36 }

38 /* Funkcija ucitava podatke o jednoj potrosackoj korpi. */
   void ucitaj_korpu(Korpa *k) {
40   int i;
     printf("Unesite podatke o korpi: \n");
42
     /* Ucitavanje broja artikala u korpi. */
44   printf("Broj artikala: ");
     scanf("%d", &k->broj_artikala);
46   if (k->broj_artikala <= 0) {
       printf("Greska: neispravan unos broja artikala (%d).\n",
48            k->broj_artikala);
       exit(EXIT_FAILURE);
50   }

52   /* Ucitavanje podataka o svakom artiklu. */
     for (i = 0; i < k->broj_artikala; i++)
54     ucitaj_artikal(&k->artikli[i]);
   }
56
   /* Funkcija ucitava podatke o n potrosackih korpi. */
58 void ucitaj_niz_korpi(Korpa korpe[], int n) {
     int i;
60   for (i = 0; i < n; i++)
       ucitaj_korpu(&korpe[i]);
62 }

64 /* Funkcija racuna ukupan racun za datu korpu. */
   float izracunaj_racun(const Korpa *k) {
66   int i;
     float racun = 0;
68
     for (i = 0; i < k->broj_artikala; i++)
70     racun += k->artikli[i].kolicina * k->artikli[i].cena;
```

```
72    return racun;
  }
74
  /* Funkcija ispisuje racun za datu korpu. */
76  void ispisi_racun(const Korpa *k) {
    int i;
78    for (i = 0; i < k->broj_artikala; i++)
      printf("\t%s %d %.2f\n", k->artikli[i].naziv,
80            k->artikli[i].kolicina, k->artikli[i].cena);
    printf("------------------------\n");
82    printf("\tukupno: %.2f\n", izracunaj_racun(k));
  }
84
  /* Funkcija ispisuje racune za sve potrosacke korpe u nizu. */
86  void ispisi_racune_za_korpe(Korpa korpe[], int n) {
    int i;
88    for (i = 0; i < n; i++) {
      printf("\nKorpa %d:\n", i);
90      ispisi_racun(&korpe[i]);
    }
92  }

94  /* Funkcija racuna prosecnu cenu potrosacke korpe za dati niz
       potrosackih korpi. */
96  float prosek(Korpa korpe[], int n) {
    int i;
98    float prosecna_cena = 0;

100   for (i = 0; i < n; i++)
      prosecna_cena += izracunaj_racun(&korpe[i]);
102
    return prosecna_cena / n;
104 }

106 int main() {
    /* Deklaracije potrebnih promenljivih. */
108   int n;
    Korpa korpe[MAKS_KORPI];
110
    /* Ucitavanje broja potrosackih korpi i provera ispravnosti
112       ulaza. */
    printf("Unesite broj potrosackih korpi:");
114   scanf("%d", &n);
    if (n < 0 || n > MAKS_KORPI) {
116     printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
118   }

120   /* Ucitavanje podataka o potrosackim korpama. */
    ucitaj_niz_korpi(korpe, n);
122
    /* Ispis svih racuna. */
```

```
  ispisi_racune_za_korpe(korpe, n);

  /* Ispis prosecne cene potrosacke korpe. */
  printf("Prosecna cena potrosacke korpe: %.2f\n",
         prosek(korpe, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.11

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAKS 50

/* Struktura koja opisuje loptu. */
typedef struct {
  int poluprecnik;
  enum { plava = 1, zuta, crvena, zelena } boja;
} Lopta;

/* Funkcija racuna zapreminu lopte. */
float zapremina(const Lopta *l) {
  return pow(l->poluprecnik, 3) * 4 / 3 * M_PI;
}

/* Funkcija racuna zbir zapremina svih lopti u nizu. */
float ukupna_zapremina(Lopta lopte[], int n) {
  int i;
  float ukupno = 0;

  for (i = 0; i < n; i++)
    ukupno += zapremina(&lopte[i]);

  return ukupno;
}

/* Funkcija broji lopte cija je boja jednaka boji koja je
   prosledjena kao argument funkcije. */
int broj_lopti_u_boji(Lopta lopte[], int n, unsigned boja) {
  int brojac = 0, i;

  for (i = 0; i < n; i++)
    if (lopte[i].boja == boja)
      brojac++;

  return brojac;
}

```

```
int main() {
  /* Deklaracije potrebnih promenljivih. */
  Lopta lopte[MAKS];
  int i, n;
  unsigned boja;

  /* Ucitavanje broja lopti i provera ispravnosti ulaza. */
  printf("Unesite broj lopti: ");
  scanf("%d", &n);
  if (n < 0 || n > MAKS) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje lopti u niz. */
  printf("Unesite poluprecnike i boje lopti "
         "(1-plava, 2-zuta, 3-crvena, 4-zelena):\n");
  for (i = 0; i < n; i++) {
    printf("%d. lopta: ", i + 1);
    scanf("%d%u", &lopte[i].poluprecnik, &boja);
    if (boja < 1 || boja > 4) {
      printf("Greska: neispravan unos.\n");
      exit(EXIT_FAILURE);
    }
    lopte[i].boja = boja;
  }

  /* Ispis rezultata. */
  printf("Ukupna zapremina: %.2f\n", ukupna_zapremina(lopte, n));
  printf("Ukupno crvenih lopti: %d\n",
         broj_lopti_u_boji(lopte, n, crvena));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.12

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAKS_TEMENA 1000

/* Struktura koja opisuje tacku. */
typedef struct {
  int x, y;
} Tacka;

/* Funkcija racuna rastojanje izmedju dve tacke. */
double rastojanje(const Tacka *a, const Tacka *b) {
  return sqrt(pow(a->x - b->x, 2) + pow(a->y - b->y, 2));
```

```
}

/* Funkcija ucitava temena poligona. */
int ucitaj_poligon(Tacka poligon[], int maks_temena) {
  int i = 0;

  printf("Unesite temena poligona:\n");
  while (scanf("%d%d", &poligon[i].x, &poligon[i].y) != EOF) {
    i++;
    if (i >= maks_temena)
      break;
  }

  return i;
}

/* Funkcija racuna obim poligona. */
double obim_poligona(Tacka poligon[], int n) {
  double obim = 0;
  int i;

  for (i = 0; i < n - 1; i++)
    obim += rastojanje(&poligon[i], &poligon[i + 1]);

  obim += rastojanje(&poligon[n - 1], &poligon[0]);

  return obim;
}

/* Funkcija racuna najduzu stranicu poligona. */
double maksimalna_stranica(Tacka poligon[], int n) {
  double maks = rastojanje(&poligon[0], &poligon[n - 1]);
  double stranica;
  int i;

  for (i = 0; i < n - 1; i++) {
    stranica = rastojanje(&poligon[i], &poligon[i + 1]);
    if (stranica > maks)
      maks = stranica;
  }

  return maks;
}

/* Funkcija racuna povrsinu trougla cija su temena A, B i C. */
double povrsina_trougla(const Tacka *A, const Tacka *B,
                        const Tacka *C) {
  double a = rastojanje(B, C);
  double b = rastojanje(A, C);
  double c = rastojanje(A, B);
  double s = (a + b + c) / 2;

```

```
  return sqrt(s * (s - a) * (s - b) * (s - c));
}

/* Funkcija racuna povrsinu poligona. */
double povrsina_poligona(Tacka *poligon, int n) {
  double povrsina = 0;
  int i;

  for (i = 1; i < n - 1; i++)
    povrsina += povrsina_trougla(&poligon[0], &poligon[i],
                          &poligon[i + 1]);

  return povrsina;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int maks_temena, n;
  Tacka poligon[MAKS_TEMENA];

  /* Ucitavanje maksimalnog broja temena i provera ispravnosti. */
  printf("Unesite maksimalan broj temena poligona: ");
  scanf("%d", &maks_temena);
  if (maks_temena < 3 || maks_temena > MAKS_TEMENA) {
    printf("Greska: neispravan unos.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje poligona. */
  n = ucitaj_poligon(poligon, maks_temena);
  if (n < 3) {
    printf("Greska: poligon mora imati bar tri temena.\n");
    exit(EXIT_FAILURE);
  }

  /* Ispis rezultata. */
  printf("Obim poligona je %.3lf.\n", obim_poligona(poligon, n));
  printf("Duzina maksimalne stranice je %.3lf.\n",
         maksimalna_stranica(poligon, n));
  printf("Povrsina poligona je %.3lf.\n",
         povrsina_poligona(poligon, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.13

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS 1000
```

```
6  /* Struktura koja opisuje izraz. */
   typedef struct {
8    char o;
     int x;
10   int y;
   } Izraz;
12
   /* Funkcija proverava da li je izraz ispravno zadat. */
14 int korektan_izraz(const Izraz *izraz) {
     if (izraz->o != '+' && izraz->o != '-' &&
16       izraz->o != '*' && izraz->o != '/') {
       printf("Greska: neispravna operacija.\n");
18     return 0;
     }
20
     if (izraz->o == '/' && izraz->y == 0) {
22     printf("Greska: deljenje nulom.\n");
       return 0;
24   }

26   return 1;
   }
28
   /* Funkcija ucitava n izraza sa standardnog ulaza. */
30 void ucitaj(Izraz izrazi[], int n) {
     int i;
32
     printf("Unesite izraze u prefiksnoj notaciji:\n");
34   for (i = 0; i < n; i++) {
       scanf("%c%d%d", &izrazi[i].o, &izrazi[i].x, &izrazi[i].y);
36     /* Preskace se novi red koji se nalazi nakon izraza, kako bi
          naredni izraz bio ispravno ucitan. */
38     getchar();

40     /* Provera ispravnosti ucitanog izraza. */
       if (!korektan_izraz(&izrazi[i])) {
42       printf("Greska: neispravan unos.\n");
         exit(EXIT_FAILURE);
44     }
     }
46 }

48 /* Funkcija racuna vrednost izraza. */
   int vrednost(const Izraz *izraz) {
50   switch (izraz->o) {
     case '+':
52     return izraz->x + izraz->y;
     case '-':
54     return izraz->x - izraz->y;
     case '*':
56     return izraz->x * izraz->y;
```

```
    case '/':
58    return izraz->x / izraz->y;
    default:
60    printf("Greska: neispravna operacija.\n");
      exit(EXIT_FAILURE);
62  }
}
64
/* Funkcija racuna najvecu vrednost izraza. */
66 int najveca_vrednost(Izraz izrazi[], int n) {
    int i, maks_vrednost, trenutna_vrednost;
68
    maks_vrednost = vrednost(&izrazi[0]);
70
    for (i = 1; i < n; i++) {
72    trenutna_vrednost = vrednost(&izrazi[i]);
      if (trenutna_vrednost > maks_vrednost)
74      maks_vrednost = trenutna_vrednost;
    }
76
    return maks_vrednost;
78 }

80 int main() {
    /* Deklaracije potrebnih promenljivih. */
82  int i, n;
    Izraz izrazi[MAKS];
84  int maks, trenutna_vrednost;
    float polovina;
86
    /* Ucitavanje broja izraza i provera ispravnosti ulaza. */
88  printf("Unesite broj izraza: ");
    scanf("%d", &n);
90  if (n < 0 || n > MAKS) {
      printf("Greska: neispravan unos.\n");
92    exit(EXIT_FAILURE);
    }
94
    /* Preskace se belina koja se unosi nakon broja izraza. Ovaj
96     korak je neophodan jer se izraz zadaje u formatu:
       <operacija> <operand> <operand>
98     Kako je <operacija> tipa char, izostavljanjem ovog koraka,
       ta belina bi bila ucitana kao <operacija> za prvi izraz. */
100 getchar();
    ucitaj(izrazi, n);
102
    /* Pronalazak polovine maksimalne vrednosti. */
104 maks = najveca_vrednost(izrazi, n);
    printf("Maksimalna vrednost izraza: %d\n", maks);
106 polovina = maks / 2.0;

108 /* Ispis rezultata. */
```

```
  printf("Izrazi cija je vrednost manja od polovine maksimalne "
         "vrednosti:\n");
  for (i = 0; i < n; i++) {
    trenutna_vrednost = vrednost(&izrazi[i]);
    if (trenutna_vrednost < polovina) {
      printf("%d %c %d = %d\n", izrazi[i].x, izrazi[i].o,
             izrazi[i].y, trenutna_vrednost);
    }
  }

  exit(EXIT_SUCCESS);
}
```

Rešenje 2.9.14

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAKS_STEPEN 10
#define MAKS_POLINOMA 100

/* Struktura koja opisuje polinom. Polinom stepena MAKS_STEPEN
   moze imati najvise MAKS_STEPEN+1 koeficijenata, a njegov
   integral onda moze imati najvise MAKS_STEPEN+2 koeficijenata. */
typedef struct {
  int stepen;
  float koef[MAKS_STEPEN + 2];
} Polinom;

/* Funkcija ucitava podatke o polinomima. */
int ucitaj(Polinom niz[]) {
  int i = 0, j;

  while (i < MAKS_POLINOMA) {
    printf("Unesite stepen: ");
    if (scanf("%d", &(niz[i].stepen)) == EOF)
      break;

    if (niz[i].stepen > MAKS_STEPEN || niz[i].stepen < 0) {
      printf("Greska: neispravan unos stepena.\n");
      exit(EXIT_FAILURE);
    }

    printf("Unesite koeficijente polinoma:\n");
    for (j = 0; j <= niz[i].stepen; j++)
      scanf("%f", &(niz[i].koef[j]));

    i++;
  }

```

```
  return i;
}

/* Prvi monom je specijalan jer se ispred njega ne vrsi eksplicitan
   ispis znaka. Na primer, za polinom x + 3*x^2, prvi monom je x.
   Svakom sledecem monomu (u ovom slucaju samo 3*x^2) u ispisu
   prethodi znak (+ ili -). Funkcija ispisuje prvi monom. */
void ispis_prvog_monoma(float koef, int stepen) {
  printf("%.2f", koef);

  if (stepen == 1)
    printf("*x ");
  else if (stepen > 1)
    printf("*x^%d ", stepen);
}

/* Funkcija ispisuje monom koji nije prvi. */
void ispis_monoma(float koef, int stepen) {
  /* Monomi ciji je koeficijent nula se ne ispisuju. */
  if (koef != 0) {
    /* Ispis znaka. */
    if (koef > 0)
      printf("+ ");
    else
      printf("- ");

    /* Ispis koeficijenta. */
    printf("%.2f", fabs(koef));

    /* Ispis ostatka. */
    if (stepen == 1)
      printf("*x ");
    else if (stepen > 1)
      printf("*x^%d ", stepen);
  }
}

/* Funkcija ispisuje ceo polinom p. */
void ispis(const Polinom *p) {
  int i;

  /* Vrsi se ispis prvog monoma. Posto je moguce da prvi monom ima
     koeficijent 0, trazi se prvi monom sa koeficijentom razlicitim
     od nule. */
  for (i = 0; i <= p->stepen; i++)
    if (p->koef[i] != 0) {
      ispis_prvog_monoma(p->koef[i], i);
      i++;
      break;
    }

  /* Ispis ostalih monoma. Nastavlja se od mesta gde se stalo u
```

```
     prethodnoj petlji i iz tog razloga je preskocen korak
     inicijalizacije brojaca i. */
  for (; i <= p->stepen; i++)
    ispis_monoma(p->koef[i], i);

  printf("\n");
}

/* Funkcija racuna integral polinoma p. */
void integral(const Polinom *p, Polinom *tekuci_integral) {
  int i;

  tekuci_integral->stepen = p->stepen + 1;
  tekuci_integral->koef[0] = 0;

  for (i = 1; i <= tekuci_integral->stepen; i++)
    tekuci_integral->koef[i] = (float) p->koef[i - 1] / i;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  Polinom polinomi[MAKS_POLINOMA], tekuci_integral;
  int n, i;

  /* Ucitavanje polinoma. */
  n = ucitaj(polinomi);

  /* Ispis integrala. */
  printf("\n\nIntegrali su:\n");
  for (i = 0; i < n; i++) {
    integral(&polinomi[i], &tekuci_integral);
    ispis(&tekuci_integral);
  }

  exit(EXIT_SUCCESS);
}
```

3

Ulaz i izlaz programa

3.1 Argumenti komandne linije

Zadatak 3.1.1 Napisati program koji ispisuje broj navedenih argumenata komandne linije, a zatim i same argumenate i njihove redne brojeve.

Primer 1

```
Pokretanje: ./a.out d1.txt 10 13.5 d2.txt

Izlaz:
  Broj argumenata je 5.
  0: ./a.out
  1: d1.txt
  2: 10
  3: 13.5
  4: d2.txt
```

Primer 2

```
Pokretanje: ./a.out

Izlaz:
  Broj argumenata je 1.
  0: ./a.out
```

Zadatak 3.1.2 Napisati program koji ispisuje zbir celobrojnih argumenata komandne linije. Uputstvo: *Koristiti funkciju* `atoi`.

Primer 1

```
Pokretanje:
  ./a.out 5 ana 9 -2 11 4 +2

Izlaz:
  Zbir celobrojnih argumenata:
  29
```

Primer 2

```
Pokretanje:
  ./a.out a1 b1 1a 1b

Izlaz:
  Zbir celobrojnih argumenata:
  0
```

Primer 3

```
Pokretanje:
  ./a.out 33 1 @matf 44 22.56

Izlaz:
  Zbir celobrojnih argumenata:
  78
```

Zadatak 3.1.3 Napisati program koji na osnovu broja n koji se zadaje kao

argument komandne linije, ispisuje cele brojeve iz intervala $[-n,\ n]$. U slučaju neispravnog pokretanja programa ispisati odgovarajuću poruku o grešci.

Primer 1

```
Pokretanje: ./a.out 2

Izlaz:
 -2 -1 0 1 2
```

Primer 2

```
Pokretanje: ./a.out

Izlaz:
 Greska: neispravan poziv.
```

Primer 3

```
Pokretanje: ./a.out 0

Izlaz:
 0
```

Zadatak 3.1.4 Napisati program koji ispisuje argumente komandne linije koji počinju karakterom *@*.

Primer 1

```
Pokretanje:
 ./a.out @ana @aca #zvezda

Izlaz:
 Argumenti koji pocinju sa @:
 @ana @aca
```

Primer 2

```
Pokretanje:
 ./a.out sanke @zapad zujanje

Izlaz:
 Argumenti koji pocinju sa @:
 @zapad
```

Primer 3

```
Pokretanje:
 ./a.out bundeva pomorandza

Izlaz:
 Nema argumenata koji
 pocinju sa @.
```

Zadatak 3.1.5 Napisati program koji ispisuje broj argumenata komandne linije koji sadrže karakter *@*.

Primer 1

```
Pokretanje:
 ./a.out pera@gmail.com @

Izlaz:
 Rezultat: 2
```

Primer 2

```
Pokretanje:
 ./a.out japan caj

Izlaz:
 0
```

Primer 3

```
Pokretanje:
 ./a.out

Izlaz:
 Rezultat: 0
```

Zadatak 3.1.6 Napisati program koji proverava da li se među zadatim argumentima komandne linije nalaze barem dva ista. Napomena: *Uzeti u obzir i naziv programa koji se pokreće.*

Primer 1

```
Pokretanje:
 ./a.out ulaz.txt izlaz.txt ulaz.txt

Izlaz:
 Medju argumentima ima istih.
```

Primer 2

```
Pokretanje:
 ./a.out srce pik tref tref

Izlaz:
 Medju argumentima ima istih.
```

Primer 3

POKRETANJE:
```
./a.out Riba ribi grize rep.
```

IZLAZ:
```
Medju argumentima nema istih.
```

Primer 4

POKRETANJE:
```
./a.out
```

IZLAZ:
```
Medju argumentima nema istih.
```

Zadatak 3.1.7 Napisati program koji ispisuje sve opcije koje su navedene u komandnoj liniji. Opcije su karakteri argumenata komandne linije koji za koje važi da počinju karakterom `-`.

Primer 1

POKRETANJE:
```
./a.out -rf in.txt
```

IZLAZ:
```
Opcije su: r f
```

Primer 2

POKRETANJE:
```
./a.out
```

IZLAZ:
```
Medju argumentima nema opcija.
```

Primer 3

POKRETANJE:
```
./a.out ulaz.txt
```

IZLAZ:
```
Medju argumentima nema opcija.
```

Primer 4

POKRETANJE:
```
./a.out in.txt -l -n 10 -fi out.txt
```

IZLAZ:
```
Opcije su: l n f i
```

3.2 Rešenja

Rešenje 3.1.1

```
#include <stdio.h>

/* Argumenti komandne linije cuvaju se u nizu niski. Svaki element
   tog niza odgovara jednom argumentu komandne linije, pri cemu
   prvi element predstavlja naziv programa koji se pokrece.
   Celobrojna promenljiva argc predstavlja ukupan broj argumenata
   komandne linije ukljucujuci i argument koji odgovara nazivu
   programa, a promenljiva argv pomenuti niz niski koji sadrzi same
   argumente. */
int main(int argc, char *argv[]) {
  /* Deklaracija potrebne promenljive. */
  int i;

  /* Ispis broja argumenata komandne linije. */
  printf("Broj argumenata je %d.\n", argc);

  /* Ispis svakog od navedenih argumenata. */
  for (i = 0; i < argc; i++)
    printf("%d: %s\n", i, argv[i]);

  return 0;
}
```

Rešenje 3.1.2

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/* Funkcija proverava da li prosledjenu nisku cine samo karakteri
   koji su cifre. */
int samo_cifre(char arg[]) {
  int i;

  /* Prvi karakter mora biti ili cifra ili znak broja. */
  if (!isdigit(arg[0]) && arg[0] != '+' && arg[0] != '-')
    return 0;

  /* Ostali karakteri moraju biti cifre. */
  for (i = 1; arg[i]; i++)
    if (!isdigit(arg[i]))
      return 0;

  return 1;
}
```

```
int main(int argc, char *argv[]) {
  /* Deklaracija potrebnih promenljivih. */
  int i, suma = 0;

  /* Kako su argumenti komandne linije niske, potrebno ih je
     konvertovati u brojeve. Za ovo je moguce koristiti funkciju
     atoi. Npr. atoi("567") ima vrednost 567. Treba voditi racuna:
     atoi("abc") ima vrednost 0, ali atoi("12abc") ima vrednost 12.
     Dakle ova funkcija se zaustavlja u trenutku kada se u okviru
     niske naidje na prvi karakter koji nije cifra. Iz tog razloga
     je potrebno proveriti da li dati argument sadrzi samo cifre. */
  for (i = 1; i < argc; i++)
    if (samo_cifre(argv[i]))
      suma += atoi(argv[i]);

  /* Ispis rezultata. */
  printf("Zbir celobrojnih argumenata: %d\n", suma);

  return 0;
}
```

Rešenje 3.1.3

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[]) {
  /* Deklaracija potrebnih promenljivih. */
  int n, i;

  /* Provera broja argumenata komandne linije. */
  if (argc != 2) {
    printf("Greska: neispravan poziv.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje broja n i cuvanje njegove apsolutne vrednosti. */
  n = atoi(argv[1]);
  n = abs(n);

  /* Ispis rezultata. */
  for (i = -n; i <= n; i++)
    printf("%d ", i);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.1.4

```
#include <stdio.h>

int main(int argc, char *argv[]) {
  /* Deklaracija potrebnih promenljivih. */
  int i, prikazi_poruku = 0;

  /* Ispis svih argumenata komandne linije ciji je prvi karakter
     znak '@'. Ako se program pokrene sa:
     ./a.out @pera mika @zika
     argv[0] je "./a.out" i on se preskace.
     argv[1] je "@pera", a prvi karakter je onda argv[1][0].
     Dakle, za argv[i] treba proveravati da li je argv[i][0] jednak
     karakteru '@'. */
  for (i = 1; i < argc; i++) {
    if (argv[i][0] == '@') {
      /* Promenljiva prikazi_poruku sluzi da detektuje da li
         postoji bar jedna niska koja pocinje sa '@'. Ukoliko se
         naidje na prvu takvu nisku, ispisuje se trazena poruka i
         prikazi_poruku se postavlja na 1. */
      if (!prikazi_poruku) {
        printf("Argumenti koji pocinju sa @:\n");
        prikazi_poruku = 1;
      }
      printf("%s ", argv[i]);
    }
  }

  /* Ukoliko je vrednost promenljive prikazi_poruku i dalje 0,
     znaci da nijedan argument ne pocinje karakterom '@'. */
  if (!prikazi_poruku)
    printf("Nema argumenata koji pocinju sa @.");
  printf("\n");

  return 0;
}
```

Rešenje 3.1.5

```
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[]) {
  /* Deklaracija potrebnih promenljivih. */
  int i, brojac = 0;

  /* Prebrojavanje argumenata koji sadrze karakter @. */
  for (i = 1; i < argc; i++)
    if (strchr(argv[i], '@') != NULL)
      brojac++;
```

```
13   /* Ispis rezultata. */
     printf("Rezultat: %d\n", brojac);
15
     return 0;
17 }
```

Rešenje 3.1.6

```
1  #include <stdio.h>
   #include <string.h>
3
   int main(int argc, char *argv[]) {
5    /* Deklaracije potrebnih promenljivih. */
     int i, j;
7
     /* Ukoliko je naveden jedan ili nijedan argument, onda ne moze da
9       bude duplikata. */
     if (argc < 2) {
11     printf("Medju argumentima nema istih.\n");
       return 0;
13   }

15   /* Za svaki argument komandne linije se proverava da li postoji
        neki od argumenata koji mu je jednak. */
17   for (i = 0; i < argc; i++) {
       /* Za fiksirano argv[i] se vrsi provera svih argumenata koji se
19        nalaze nakon njega. */
       for (j = i + 1; j < argc; j++)
21       if (strcmp(argv[i], argv[j]) == 0) {
           printf("Medju argumentima ima istih.\n");
23         return 0;
         }
25   }

27   /* Ukoliko se prethodna petlja zavrsila, a nije se izaslo iz
        programa, znaci da medju argumentima nema istih. */
29   printf("Medju argumentima nema istih.\n");

31   return 0;
   }
```

Rešenje 3.1.7

```
1  #include <stdio.h>

3  int main(int argc, char *argv[]) {
     /* Deklaracija potrebnih promenljivih. */
5    int i, j, prikazi_poruku = 0;
```

```
7   /* Prolazi se kroz sve argumente komandne linije. */
    for (i = 1; i < argc; i++) {
9     /* Ukoliko argument pocinje karakterom '-', znaci da se navode
         opcije. */
11    if (argv[i][0] == '-') {
        /* Ukoliko je u pitanju prvi niz opcija, ispisuje se
13         odgovarajuca poruka i vrednost promenljive prikazi_poruku
           se postavlja na 1. */
15      if (!prikazi_poruku) {
          printf("Opcije su: ");
17        prikazi_poruku = 1;
        }
19
        /* Ispisuju se sve opcije, tj. svi karakteri argumenta
21         argv[i] koji se nalaze nakon '-'. */
        for (j = 1; argv[i][j]; j++)
23        printf("%c ", argv[i][j]);
      }
25  }

27  /* Ukoliko je vrednost promenljive prikazi_poruku nakon petlje 0,
       znaci da nije navedena nijedna opcija. */
29  if (!prikazi_poruku)
      printf("Medju argumentima nema opcija.\n");
31  else
      printf("\n");
33
    return 0;
35  }
```

3.3 Datoteke

Zadatak 3.3.1 Napisati program koji prepisuje sadržaj datoteke *ulaz.txt* u datoteku *izlaz.txt* karakter po karakter. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
ULAZ.TXT
 Danas je 21. mart.
 To je prvi dan proleca.

IZLAZ.TXT
 Danas je 21. mart.
 To je prvi dan proleca.
```

Primer 2

```
ULAZ.TXT
 Ispit iz Programiranja 1 je
 zakazan za 10. jun.

IZLAZ.TXT
 Ispit iz Programiranja 1 je
 zakazan za 10. jun.
```

Primer 3

```
ULAZ.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
 Greska: neuspesno otvaranje
 datoteke ulaz.txt
```

Zadatak 3.3.2 Napisati program koji prepisuje svaki treći karakter datoteke *ulaz.txt* u datoteku *izlaz.txt*. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
ULAZ.TXT
 Volim programiranje.

IZLAZ.TXT
 Vipgmae
```

Primer 2

```
ULAZ.TXT
 Zivot je lep!

IZLAZ.TXT
 Zojl!
```

Primer 3

```
ULAZ.TXT
 1234567890

IZLAZ.TXT
 1470
```

Primer 4

```
ULAZ.TXT
 Ova datoteka
 sadrzi tekst
 u vise
 linija.

IZLAZ.TXT
 O te
 diet sli.
```

Primer 5

```
ULAZ.TXT
 U Beogradu ce biti
 suncan i lep
 dan.

IZLAZ.TXT
 Ueruei
 nn pa
```

Zadatak 3.3.3 Napisati program koji šifruje sadržaj datoteke *podaci.txt* tako što svako slovo ciklično zamenjuje njegovim prethodnikom suprotne veličine i upisuje u datoteku *sifra.txt*. Na primer, slovo `b` se zamenjuje slovom `A`, slovo `B` slovom `a`, slovo `a` slovom `Z`, slovo `A` slovom `z`, itd. Ostali karakteri ostaju nepromenjeni. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
PODACI.TXT
  Matematicki fakultet
  Studentski trg 16
  Beograd

SIFRA.TXT
  lZSDLZSHBJH EZJTKSDS
  rSTCDMSRJH SQF 16
  aDNFQZC
```

Primer 2

```
PODACI.TXT
  a=x+y;
  x=b+5;

SIFRA.TXT
  Z=W+X;
  W=A+5;
```

Primer 3

```
PODACI.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
  Greska: neuspesno otvaranje
  datoteke podaci.txt.
```

Zadatak 3.3.4 Napisati program koji za dve datoteke čija se imena unose sa standarnog ulaza, radi sledeće:

- za svaku cifru u prvoj datoteci, u drugu datoteku upisuje 0
- za svako slovo u prvoj datoteci, u drugu datoteku upisuje 1
- za sve ostale karaktere u prvoj datoteci, u drugu datoteku upisuje 2

Pretpostaviti da je maksimalna dužina naziva datoteka 20 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime prve datoteke:
  kilometraze.txt
  Unesite ime druge datoteke:
  sifra.txt

KILOMETRAZE.TXT
  Beograd - Nis 230km
  Uzice - Cacak 56.3km
  Subotica - Ruma 139km

SIFRA.TXT
  11111112221112000112111112 2
  21111120020112111111122211
  11200011
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime prve datoteke:
  ulaz.txt
  Unesite ime druge datoteke:
  izlaz.txt

ULAZ.TXT
  18. februar 2019.

IZLAZ.TXT
  00221111111200002
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime prve datoteke:
  in.dat
  Unesite ime druge datoteke:
  out.dat

IN.DAT NE POSTOJI

IZLAZ ZA GREŠKE:
  Greska: neuspesno otvaranje
  prve datoteke.
```

Zadatak 3.3.5 Sa standarnog ulaza učitavaju se imena dveju datoteka i jedan karakter koji označava opciju. Napisati program koji prepisuje sadržaj prve datoteke u drugu tako što u slučaju da je navedena opcija `u`, sva mala slova zamenjuje velikim slovima, a u slučaju da je navedena opcija `l`, sva velika slova zamenjuje malim slovima. Pretpostaviti da je maksimalna dužina naziva datoteka 20 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite imena datoteka i opciju:
 ulaz.txt izlaz.txt u

ULAZ.TXT
 danas je lep dan
 i Ja zelim
 da postanem programer

IZLAZ.TXT
 DANAS JE LEP DAN
 I JA ZELIM
 DA POSTANEM PROGRAMER
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite imena datoteka i opciju:
 prva.dat druga.dat l

PRVA.DAT
 Cena soka je 30
 Cena vina je 150
 Cena limunade je 200
 Cena sendvica je 120

DRUGA.DAT
 cena soka je 30
 cena vina je 150
 cena limunade je 200
 cena sendvica je 120
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite imena datoteka i opciju:
 primer.c prazna.txt V

PRIMER.C
 #include <stdio.h>
 int main()
 {
 }

PRAZNA.TXT


IZLAZ ZA GREŠKE:
 Greska: neispravan unos.
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite imena datoteka i opciju:
 primer.c prazna.txt V

PRIMER.C NE POSTOJI

IZLAZ ZA GREŠKE:
 Greska: neuspesno otvaranje
 prve datoteke.
```

Zadatak 3.3.6 Napisati program koji prebrojava mala slova u datoteci *podaci.txt* i dobijeni rezultat ispisuje na standardni izlaz. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
PODACI.TXT
 Matematicki fakultet
 Studentski trg 16
 Beograd

IZLAZ:
 Broj malih slova je: 36
```

Primer 2

```
PODACI.TXT
 PrograMiranje

IZLAZ:
 Broj malih slova je: 11
```

Primer 3

```
PODACI.TXT
 MATEMATIKA
 12+34=46

IZLAZ:
 Broj malih slova je: 0
```

Zadatak 3.3.7 Napisati program koji u datoteci čije se ime unosi sa standardnog ulaza prebrojava koliko se puta pojavljuje svaka od cifara i na standardni izlaz ispisuje cifru sa najvećim brojem pojavljivanja. Ukoliko ima više takvih ci-

fara, ispisati sve. Ukoliko datoteka ne sadrži nijednu cifru, ispisati odgovarajuću poruku. Pretpostaviti da je maksimalna dužina naziva datoteke 20 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  ulaz.txt

ULAZ.TXT
  danas je lep dan
  i ja zelim
  da postanem programer

IZLAZ:
  Datoteka ne sadrzi cifre.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  prva.dat

PRVA.DAT
  Cena soka je 30
  Cena vina je 150
  Cena limunade je 200
  Cena sendvica je 120

IZLAZ:
  Najcesce cifre: 0
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  primer.c

PRIMER.C
  1 22 333.444

IZLAZ:
  Najcesce cifre: 3 4
```

Zadatak 3.3.8 Napisati program koji u datoteci čije je ime dato kao argument komandne linije proverava da li su zagrade pravilno uparene. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE:
  ./a.out zagrade.txt

ZAGRADE.TXT
  ab( cd) ..
  ((3+4)*5+1)*9

IZLAZ:
  Zagrade jesu uparene.
```

Primer 2

```
POKRETANJE:
  ./a.out primer2.dat

PRIMER2.DAT
  (7+8
  nisu(
  uparene

IZLAZ:
  Zagrade nisu uparene.
```

Primer 3

```
POKRETANJE:
  ./a.out primer3.dat

PRIMER3.DAT
  )) 7 + 6 ((

IZLAZ:
  Zagrade nisu uparene.
```

Zadatak 3.3.9 Napisati program koji prebrojava slova i cifre u datoteci.

(a) Napisati funkciju `int ucitaj_karaktere(char s[], FILE *f)` kojom se učitavaju karakteri iz datoteke `f` u niz karaktera `s`. Dozvoljeni karakteri za učitavanje su mala i velika slova engleske abecede, kao i cifre. Učitavanje se prekida kada se naiđe na znak za novi red ili nedozvoljeni karakter. Funkcija vraća broj elemenata niza uspešno učitanih karaktera.

(b) Napisati funkciju `void prebroj(char s[], int n, int *broj_slova, int *broj_cifara)` kojom se određuje broj slovnih elemenata niza karaktera (velikih ili malih slova) kao i broj cifara.

Napisati program koji koristeći prethodne funkcije prebrojava cifre i slova u datoteci čije se ime zadaje kao argument komandne linije, a zatim ispisuje dobijene

vrednosti na standardni izlaz. Pretpostaviti da je maksimalni broj karaktera datoteke 1000. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE:
  ./a.out skup.txt

SKUP.TXT
  OvoJeSkupKaraktera.123

IZLAZ:
  Broj slova: 18
  Broj cifara: 0
```

Primer 2

```
POKRETANJE:
  ./a.out skup2.txt

SKUP2.TXT
  ovdeimamo$dolar

IZLAZ:
  Broj slova: 9
  Broj cifara: 0
```

Primer 3

```
POKRETANJE:
  ./a.out skup3.txt

SKUP3.TXT
  broj3
  broj5

IZLAZ:
  Broj slova: 4
  Broj cifara: 1
```

Primer 4

```
POKRETANJE:
  ./a.out skup4.txt

SKUP4.TXT
  11.2.2019.

IZLAZ:
  Broj slova: 0
  Broj cifara: 2
```

Primer 5

```
POKRETANJE:
  ./a.out skup5.txt

SKUP5.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
  Greska: neuspesno otvaranje
  ulazne datoteke.
```

Primer 6

```
POKRETANJE:
  ./a.out

IZLAZ ZA GREŠKE:
  Greska: neispravan poziv.
```

Zadatak 3.3.10 Napisati program koji sa standardnog ulaza učitava reč s i u datoteku *rotacije.txt* upisuje sve njene rotacije. Pretpostaviti da je maksimalna dužina reči 20 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
  Unesite rec: abcde

ROTACIJE.TXT
  abcde
  bcdea
  cdeab
  deabc
  eabcd
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
  Unesite rec: 1234

ROTACIJE.TXT
  1234
  2341
  3412
  4123
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
  Unesite rec: a=3*x+5;

ROTACIJE.TXT
  a=3*x+5;
  =3*x+5;a
  3*x+5;a=
  *x+5;a=3
  x+5;a=3*
  +5;a=3*x
  5;a=3*x+
  ;a=3*x+5
```

Zadatak 3.3.11 Sa standarnog ulaza se učitava ime datoteke i nenegativan ceo broj k. Napisati program koji učitava reči iz datoteke (reč je niz karaktera

između blanko simbola) i svaku pročitanu reč rotira za k mesta u levo i tako dobijenu reč upisuje u datoteku čije je ime *rotirano.txt*. Pretpostaviti da je maksimalna dužina naziva datoteke 20 karaktera, da datoteka sadrži samo slova i beline i da je maksimalna dužina jedne reči u datoteci 100 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  ulaz.txt
  Unesite broj k: 3

ULAZ.TXT
  jedan dva
  tri cetiri

ROTIRANO.TXT
  anjed dva tri iricet
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  in.dat
  Unesite broj k: 5

IN.DAT
  Popodne ce biti kise

ROTIRANO.TXT
  nePopod ec itib isek
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  input.txt
  Unesite broj k: 0

INPUT.TXT
  Popodne ce
  biti kise

ROTIRANO.TXT
  Popodne ce biti kise
```

Zadatak 3.3.12 Napisati program koji iz datoteke *razno.txt* u datoteku *palindromi.txt* prepisuje sve palindrome. Reč je palindrom ako se isto čita sa leve i desne strane bez obzira na veličinu slova. Pretpostaviti da je maksimalna dužina reči 200 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
RAZNO.TXT
  Ana i melem su
  primeri palindroma.

PALINDROMI.TXT:
  Ana i melem
```

Primer 2

```
RAZNO.TXT
  jabuka neven pomorandza
  kuk Oko kapAk pero radar
  caj

PALINDROMI.TXT:
  neven kuk Oko
  kapAk radar
```

Primer 3

```
RAZNO.TXT
  ovde nema palindroma

PALINDROMI.TXT:
```

Zadatak 3.3.13 U datoteci čije se ime zadaje sa standardnog ulaza nalazi se broj n ($n \leq 256$), a zatim i n reči. Napisati program koji učitava reči iz datoteke u niz i iz niza uklanja sve duplikate i upisuje izmenjeni niz u datoteku *bez_duplikata.txt* Pretpostaviti da je maksimalna dužina naziva datoteke 20 karaktera, a maksimalna dužina jedne reči u datoteci 50 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  imena.txt

IMENA.TXT
  12
  Ana Milos Ana Marko
  Petar Filip Jovana Ana
  Petar Ivan Nikola Filip

BEZ_DUPLIKATA.TXT:
  Ana Milos Marko Petar
  Filip Jovana Ivan Nikola
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  gradovi.txt

GRADOVI.TXT
  10
  Sombor Beograd
  Nis Beograd
  Beograd Indjija
  Nis Ruma
  Ruma Sombor

BEZ_DUPLIKATA.TXT:
  Sombor Beograd Nis
  Indjija Ruma
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
  Unesite ime datoteke:
  gradovi.txt

GRADOVI.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
  Greska: neuspesno otvaranje
  ulazne datoteke.
```

Zadatak 3.3.14 U datoteci čije se ime zadaje kao prvi argument komandne linije nalazi se ceo pozitivan broj n, a zatim i n celih brojeva. Napisati program koji na standardni izlaz ispisuje koliko k-tocifrenih brojeva postoji u datoteci, pri čemu se pozitivan ceo broj k zadaje kao drugi argument komandne linije. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE:
  ./a.out ulaz.txt 2

ULAZ.TXT
  6
  15 193 -27 9790 35 1

IZLAZ:
  Broj 2-cifrenih brojeva: 3
```

Primer 2

```
POKRETANJE:
  ./a.out ulaz.txt 5

ULAZ.TXT
  4
  15 193 -27 9790

IZLAZ:
  Broj 5-cifrenih brojeva: 0
```

Primer 3

```
POKRETANJE:
  ./a.out ulaz.txt

IZLAZ ZA GREŠKE:
  Greska: neispravan poziv.
```

Zadatak 3.3.15 Napisati program koji na standardni izlaz ispisuje maksimum brojeva iz datoteke *brojevi.txt*. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
BROJEVI.TXT
  2.36 -16.11 5.96 8.88
  -265.31 54.96 38.4

IZLAZ:
  Najveci broj je: 54.96
```

Primer 2

```
BROJEVI.TXT
  10.5 183.111 -90.2 3.167

IZLAZ:
  Najveci broj je: 183.111
```

Primer 3

```
BROJEVI.TXT
  -62.7 -190.2 -2.3 -1000
  -198.25 -8

IZLAZ:
  Najveci broj je: -2.3
```

Zadatak 3.3.16 Prvi red datoteke *matrica.txt* sadrži dva cela broja manja od 50 koji predstavljaju redom broj vrsta i broj kolona realne matrice a. Svaki sledeći red sadrži po jednu vrstu matrice. Napisati program koji pronalazi sve elemente matrice a koji su jednaki zbiru svih svojih susednih elemenata i štampa ih u obliku `(broj vrste, broj kolone, vrednost elementa)`. Pretpostaviti da je sadržaj datoteke ispravan. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku. UPUTSTVO: *U zadatku 2.7.6 je dato objašnjenje koji elementi matrice su susedni.*

Primer 1

```
MATRICA.TXT
 3 4
 1 2 3 4
 7 2 15 -3
 -1 3 1 3

IZLAZ:
 (1, 0, 7)
 (1, 2, 15)
```

Primer 2

```
MATRICA.TXT
 2 2
 1 1
 -2 2

IZLAZ:
 (0, 0, 1)
 (0, 1, 1)
```

Primer 3

```
MATRICA.TXT
 1 4
 9 3 5 2

IZLAZ:
 (0, 2, 5)
```

Zadatak 3.3.17 Prvi red datoteke *ulaz.txt* sadrži dva cela broja između 2 i 50 koji predstavljaju redom broj vrsta i broj kolona realne matrice a. Svaki sledeći red sadrži po jednu vrstu matrice. Napisati program koji nalazi i štampa sve četvorke oblika `(a(i,j), a(i+1,j), a(i,j+1),a(i+1,j+1))` u kojima su svi elementi međusobno različiti. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
ULAZ.TXT
 3 4
 1 2 3 4
 7 2 15 -3
 -1 3 1 3

IZLAZ:
 (3, 15, 4, -3)
 (7, -1, 2, 3)
 (2, 3, 15, 1)
 (15, 1, -3, 3)
```

Primer 2

```
ULAZ.TXT
 1 4
 9 3 5 2

IZLAZ ZA GREŠKE:
 Greska: neispravna
 dimenzija.
```

Primer 3

```
ULAZ.TXT
 2 2
 1 1
 -2 2

IZLAZ:
```

Zadatak 3.3.18 U datoteci *tacke.txt* se nalazi broj tačaka, a zatim u posebnim redovima za svaku tačku njene x i y koordinate. Napisati program koji u datoteku *rastojanja.txt* upisuje rastojanje svake od učitanih tačaka od koordinatnog početka, a na standardni izlaz koordinate tačke koja je od njega najudaljenija. Ukoliko ima više takvih tačaka, ispisati koordinate prve. Koristiti

strukturu `Tacka` sa poljima `x` i `y`, kao i funkciju kojom se računa rastojanje tačke od koordinatnog početka. Pretpostaviti da je maksimalan broj tačaka u datoteci 50. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
TACKE.TXT
 4
 11 -2
 3 5
 8 -8
 0 4

RASTOJANJA.TXT
 11.18
 5.83
 11.31
 4.00

IZLAZ:
 Najudaljenija tačka: (8, -8)
```

Primer 2

```
TACKE.TXT
 -2
 0 0
 9 -8

IZLAZ ZA GREŠKE:
 Greska: neispravan broj tacaka.
```

Zadatak 3.3.19 Definisati strukturu kojom se opisuje trodimenzioni vektor sa celobrojnim koordinatama x, y i z. U datoteci *vektori.txt* nalazi se nepoznati broj vektora. Napisati program koji učitava vektore iz ove datoteke i na standardni izlaz ispisuje koordinate vektora sa najvećom dužinom. Ukoliko ima više takvih vektora, ispisati koordinate prvog. Dužina vektora se izračunava po formuli: $|v| = \sqrt{x^2 + y^2 + z^2}$. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
VEKTORI.TXT
 4 -1 7
 3 1 2

IZLAZ:
 4 -1 7
```

Primer 2

```
VEKTORI.TXT
 4 -4 4
 -4 -4 -4

IZLAZ:
 4 -4 4
```

Primer 3

```
VEKTORI.TXT
 0 0 0
 0 1 0
 1 0 0

IZLAZ:
 0 1 0
```

Primer 4

```
VEKTORI.TXT
 3 0 1
 4 5 2
 1 0 0
 2 -1 2

IZLAZ:
 4 5 2
```

Primer 5

```
VEKTORI.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
 Greska: neuspesno otvaranje
 ulazne datoteke.
```

Primer 6

```
VEKTORI.TXT
 1 1 1

IZLAZ:
 1 1 1
```

Zadatak 3.3.20 Definisati strukturu `Pravougaonik` koja sadrži dužine stranica i ime pravougaonika. Napisati program koji iz datoteke čije ime se zadaje kao argument komandne linije učitava podatke o pravougaonicima (nije poznato koliko), a zatim ispisuje imena onih pravougaonika koji su kvadrati i vrednost najveće površine među pravougaonicima koji nisu kvadrati. Pretpostaviti da je maksimalan broj pravougaonika 200, a maksimalna dužina imena pravougaonika 4. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE:
  ./a.out pravougaonici.dat

PRAVOUGAONICI.DAT
  2 4 p1
  3 3 p2
  1 6 p3

IZLAZ:
  p2 8
```

Primer 2

```
POKRETANJE:
  ./a.out dva.dat

DVA.DAT
  5 2 pm
  4 7 pv

IZLAZ:
  28
```

Primer 3

```
POKRETANJE:
  ./a.out tri.dat

TRI.DAT
  5 5 m
  3 3 s
  8 8 xl

IZLAZ:
  m s xl
```

Zadatak 3.3.21 U datoteci *studenti.txt* se nalaze podaci o studentima. Za svakog studenta je dato korisničko ime na Alas serveru i poslednjih pet ocena koje je dobio. Napisati program koji pronalazi studenta koji je ostvario najbolji uspeh i ispisuje njegove podatke. Ukoliko ima više takvih studenata, ispisati informacije o svima. Pretpostaviti da je maksimalni broj studenta 100. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
STUDENTI.TXT
  mr15239 10 9 9 8 10
  mi14005 8 8 9 8 10
  ml15112 9 8 8 7 10
  mr15007 10 10 10 10 10
  mn13208 7 7 9 6 10

IZLAZ:
  Studenti sa najvecim prosekom:
  Korisnicko ime: mr15007
  Prosek ocena: 10.00
```

Primer 2

```
STUDENTI.TXT
  mr16156 10 9 9 10 10
  mi17234 9 9 10 10 10
  ml17084 9 8 8 8 8

IZLAZ:
  Studenti sa najvecim prosekom:
  Korisnicko ime: mr16156
  Prosek ocena: 9.6

  Korisnicko ime: mi17234
  Prosek ocena: 9.6
```

Zadatak 3.3.22 Definisati strukturu `Student` koja sadrži puno ime studenta, niz njegovih ocena, broj ocena i prosečnu ocenu. U datoteci čije se ime zadaje kao argument komandne linije se nalaze podaci o studentima. Za svakog studenta dato je ime, prezime i niz ocena koji se završava nulom. Svi podaci su razdvojeni razmacima. Napisati program koji učitava podatke o studentima i na standardni izlaz ispisuje podatke za studenta sa najvećim prosekom (prosek

ispisati na 2 decimale). Ukoliko ima više takvih studenata, ispisati informacije o prvom studentu. Pretpostaviti da je maksimalni broj ocena 10 i maksimalna dužina punog imena 100 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku. UPUTSTVO: *Ime i prezime studenta se mogu pročitati pomoću specifikatora %s a potom se za kreiranje niske* `puno_ime` *u traženom formatu može iskoristiti funkcija* `strcat`.

Primer 1

```
POKRETANJE: ./a.out studenti.txt

STUDENTI.TXT
  Marko Markovic 5 6 7 8 9 0
  Jelena Jankovic 10 10 10 0
  Filip Viskovic 10 9 8 7 6 0
  Jana Peric 10 10 9 9 8 8 7 0

IZLAZ:
  Jelena Jankovic 10 10 10 10.00
```

Primer 2

```
POKRETANJE: ./a.out

IZLAZ ZA GREŠKE:
  Greska: neispravan poziv.
```

Zadatak 3.3.23 Imena ulazne i izlazne datoteke se redom navode kao argumenti komandne linije. U ulaznoj datoteci se nalaze podaci o razlomcima: u prvom redu se nalazi broj razlomaka, a u svakom sledećem redu brojilac i imenilac po jednog razlomka. Definisati strukturu koja opisuje razlomak i napisati program koji učitava niz razlomaka iz datoteke, a potom:

(a) ukoliko je prilikom pokretanja programa navedena opcija `x`, upisati u izlaznu datoteku recipročni razlomak za svaki razlomak iz niza

(b) ukoliko je prilikom pokretanja programa navedena opcija `y`, upisati u izlaznu datoteku realnu vrednost recipročnog razlomka svakog razlomka iz niza

Pretpostaviti da se u ulaznoj datoteci nalazi najviše 100 razlomaka. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE:
  ./a.out d1.txt d2.txt -x

D1.TXT
  4
  1 5
  19 3
  -2 7
  97 90
```

Primer 1 (nastavak)

```
D2.TXT
  5/1
  3/19
  -7/2
  90/97
```

Primer 2

```
POKRETANJE:
  ./a.out ulaz.txt izlaz.txt

IZLAZ ZA GREŠKE:
  Greska: neispravan poziv.
```

Primer 3

```
Pokretanje:
  ./a.out d1.txt d2.txt -y

d1.txt
  4
  1 5
  19 3
  -2 7
  97 90
```

Primer 3 (nastavak)

```
d2.txt
  5.000000
  0.157894
  -3.500000
  0.927835
```

Zadatak 3.3.24 Definisati strukturu `Automobil` koja sadrži marku, model i cenu. Napisati program koji iz datoteke čije se ime zadaje sa standardnog ulaza učitava broj automobila i podatke za svaki automobil i zatim:

(a) ispisuje prosečnu cenu po marki automobila

(b) za maksimalnu cenu koju je kupac spreman da plati, a koja se zadaje kao argument komandne linije, ispisuje automobile u tom cenovnom rangu

Pretpostaviti da se model i marka sastoje od jedne reči, da svaka od njih sadrži najviše 30 karaktera i da se u datoteci nalaze podaci za najviše 100 automobila. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
Pokretanje: ./a.out 5000

Interakcija sa programom:
  Unesite naziv datoteke:
  dat1.txt

dat1.txt ne postoji

Izlaz za greške:
  Greska: neuspesno otvaranje
  ulazne datoteke.
```

Primer 2

```
Pokretanje: ./a.out

Izlaz za greške:
  Greska: neispravan poziv.
```

Primer 3

```
Pokretanje: ./a.out 4000

Interakcija sa programom:
  Unesite naziv datoteke:
  dat1.txt

dat1.txt
  7
  renault twingo 2900
  renault megan 6250
  renault clio 3650
  dacia logan 5400
  dacia sandero 7800
```

Primer 3 (nastavak)

```
  fiat bravo 4900
  fiat linea 4290

Izlaz:
  Informacije o prosecnoj
  ceni po markama:
  renault 4266.67
  dacia 6600.00
  fiat 4595.00

  Kola u Vasem cenovnom rangu:
  renault twingo 2900
  renault clio 3650
```

Zadatak 3.3.25 Kao argumenti komandne linije zadaju se ime datoteke i ceo broj k. Napisati program koji na standardni izlaz ispisuje sve linije zadate datoteke čija je dužina veća od k. Pretpostaviti da je maksimalna dužina linije 80 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE: ./a.out test.txt 7

TEST.TXT
  Teme koje su obradjivane:
  Petlje
  Funkcije
  Nizovi
  Strukture

IZLAZ:
  Teme koje su obradjivane:
  Funkcije
  Strukture
```

Primer 2

```
POKRETANJE: ./a.out test.txt

IZLAZ ZA GREŠKE:
  Greska: neispravan poziv.
```

Zadatak 3.3.26 Napisati program koji u datoteci čije se ime navodi kao argument komandne linije određuje liniju maksimalne dužine i ispisuje je na standardni izlaz. Ukoliko ima više takvih linija, ispisati onu koja je leksikografski prva. Pretpostaviti da je maksimalna dužina linije 80 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
POKRETANJE: ./a.out test.txt

TEST.TXT
  Danas je veoma hladno decembarsko
  popodne. Ne pada sneg, kazu mozda
  ce sutra.

IZLAZ:
  Danas je veoma hladno decembarsko
```

Primer 2

```
POKRETANJE: ./a.out in.txt

IN.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
  Greska: neuspesno otvaranje
  ulazne datoteke.
```

Zadatak 3.3.27 U datoteci čije se ime navodi kao prvi argument komandne linije navedena je reč r i niz linija. Napisati program koji u datoteku čije se ime navodi kao drugi argument komandne linije upisuje sve linije prve datoteke u kojima se reč r pojavljuje bar n puta gde je n pozitivan ceo broj koji se unosi sa standardnog ulaza. Prilikom prebrojavanja, računaju se i samostalna pojavljivanja reči r i pojavljivanja u okviru neke druge reči. Ispis treba da bude u formatu `broj_pojavljivanja:linija`. Pretpostaviti da je maksimalna dužina reči 100 karaktera, a linije 500 karaktera. U slučaju greške, na standardni izlaz za greške

ispisati odgovarajuću poruku.

Primer 1

```
Pokretanje: ./a.out input.txt output.txt

Interakcija sa programom:
 Unesite broj n: 2

input.txt
 sto
 stolica lampa
 postotak Stopiranje stopa
 presto Ostoja stotina prostorija

output.txt
 2: postotak Stopiranje stopa
 4: presto Ostoja stotina prostorija
```

Primer 2

```
Pokretanje: ./a.out input.txt output.txt

Interakcija sa programom:
 Unesite broj n: 3

input.txt
 red
 redar za ovu nedelju
 redosled ured
 odrediti raspored

output.txt
```

Primer 3

```
Pokretanje: ./a.out in.txt out.txt

in.txt ne postoji

Izlaz za greške:
 Greska: neuspesno otvaranje
 ulazne datoteke.
```

Primer 4

```
Pokretanje: ./a.out in.txt

Izlaz za greške:
 Greska: neispravan poziv.
```

Zadatak 3.3.28 Napisati program koji prebrojava koliko se linija datoteke *ulaz.txt* završava niskom *s* koja se učitava sa standardnog ulaza. Pretpostaviti da je maksimalna dužina linije 80 karaktera, a niske *s* 20 karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
ulaz.txt
 /home/korisnik/imena.txt
 /home/korisnik/a.out
 /home/cv.pdf
 /home/korisnik/ulaz.txt
 /home/rezultati.xlsx
 /var/log/apache2/error.log

Interakcija sa programom:
 Unesite nisku s: .txt
 Broj linija: 2
```

Primer 2

```
ulaz.txt
 /var/log/apache2/error.log
 /var/log/dpkg.log
 moj_log.log
 /home/korisnik.login
 /home/korisnik.log.txt

Interakcija sa programom:
 Unesite nisku s: .log
 Broj linija: 3
```

Zadatak 3.3.29 Napisati program koji linije koje se učitavaju sa standardnog ulaza sve do kraja ulaza prepisuje u datoteku *izlaz.txt* i to, ako je prilikom pokretanja zadata opcija `-v` ili `-V` samo one linije koje počinju velikim slovom, ako je zadata opcija `-m` ili `-M` samo one linije koje počinju malim slovom, a ako je opcija izostavljena sve linije. Pretpostaviti da je maksimalna dužina linije 80

karaktera. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
Pokretanje: ./a.out -m

Interakcija sa programom:
  Unesite recenice:
  programiranje u C-u je zanimljivo
  Volim programiranje!
  Kada porastem bicu programer!
  u slobodno vreme programiram

izlaz.txt
  programiranje u C-u je zanimljivo
  u slobodno vreme programiram
```

Primer 2

```
Pokretanje: ./a.out -V

Interakcija sa programom:
  Unesite recenice:
  programiranje u C-u je zanimljivo
  Volim programiranje!
  Kada porastem bicu programer!
  u slobodno vreme programiram

izlaz.txt
  Volim programiranje!
  Kada porastem bicu programer!
```

Primer 3

```
Pokretanje: ./a.out -k

Izlaz za greške:
  Greska: neispravna opcija.
```

Primer 4

```
Pokretanje: ./a.out

Izlaz za greške:
  Greska: neispravan poziv.
```

Zadatak 3.3.30 Napisati program koji poredi dve datoteke i ispisuje redni broj linija u kojima se datoteke razlikuju. Imena datoteka se zadaju kao argumenti komandne linije. Pretpostaviti da je maksimalna dužina linije 200 karaktera. Linije brojati počevši od 1. U slučaju greške, na standardni izlaz za greške ispisati odgovarajuću poruku.

Primer 1

```
Pokretanje:
  ./a.out ulaz.txt izlaz.txt

ulaz.txt
  danas vezbamo
  programiranje
  ovo je primer kad su
  datoteke iste

izlaz.txt:
  danas vezbamo
  programiranje
  ovo je primer kad su
  datoteke iste

Izlaz:
```

Primer 2

```
Pokretanje:
  ./a.out u1.dat u2.dat

u1.dat
  danas vezbamo
  analizu
  ovo je primer kad
  su datoteke razlicite

u2.dat
  danas vezbamo
  programiranje
  ovo je primer kad su
  datoteke razlicite

Izlaz:
  2 3 4
```

Primer 3

```
Pokretanje:
  ./a.out prva.dat druga.dat

prva.dat
  ovo je primer
  kada su
  datoteke
  razlicite duzine

druga.dat
  ovo je primer kada
  su
  datoteke
  razlicite
  duzine
  i kada treba ispisati broj
  tih redova

Izlaz:
  1 2 4 5 6 7
```

3.4 Rešenja

Rešenje 3.3.1

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int c;

  /* Promenljive ulaz i izlaz predstavljaju pokazivace na ugradjenu
     strukturu FILE. Unutar ove strukture se nalaze polja neophodna
     za rad sa datotekama. */
  FILE *ulaz, *izlaz;

  /* Funkcija fopen sluzi da otvori datoteku. Prvi argument je
     putanja do datoteke koja se otvara, a drugi argument je niska
     koja moze imati vrednosti "r", "r+", "w", "w+", "a", "a+".
     Kada ovaj argument ima vrednost "r" datoteka se otvara za
     citanje. Ukoliko datoteka ne postoji, funkcija fopen kao
     povratnu vrednost vraca NULL. */
  ulaz = fopen("ulaz.txt", "r");
  if (ulaz == NULL) {
    /* Funkcija fprintf vrsi ispis u datoteku. Funkcionise isto kao
       i funkcija printf - razlika je sto se kao prvi argument
       prosledjuje datoteka u koju se ispisuje izlaz.

       Ukoliko je izlaz potrebno ispisati na standardni izlaz za
       greske, kao prvi argument se navodi stderr. */
    fprintf(stderr, "Greska: neuspesno otvaranje datoteke "
            "ulaz.txt.\n");
    exit(EXIT_FAILURE);
  }

  /* Ukoliko je drugi argument funkcije fopen "w", tada se
     prosledjena datoteka otvara za pisanje. */
  izlaz = fopen("izlaz.txt", "w");
  if (izlaz == NULL) {
    fprintf(stderr, "Greska: neuspesno otvaranje datoteke "
            "izlaz.txt. \n");
    exit(EXIT_FAILURE);
  }

  /* Funkcija fgetc ucitava jedan karakter iz datoteke ulaz.
     Povratna vrednost ove funkcije je ASCII kod unetog karaktera.
     Funkcija fputc ispisuje karakter c u datoteku izlaz. */
  while ((c = fgetc(ulaz)) != EOF)
    fputc(c, izlaz);

  /* Nakon zavrsetka rada sa datotekama, neophodno ih je zatvoriti
```

```
     pomocu ugradjene funkcije fclose. */
49   fclose(ulaz);
     fclose(izlaz);
51
     exit(EXIT_SUCCESS);
53 }
```

Rešenje 3.3.2

```
   #include <stdio.h>
2  #include <stdlib.h>

4  int main() {
     /* Deklaracije potrebnih promenljivih. */
6    FILE *ulaz, *izlaz;
     int c;
8
     /* Otvaranje datoteke ulaz.txt za citanje i provera uspeha. */
10   ulaz = fopen("ulaz.txt", "r");
     if (ulaz == NULL){
12     fprintf(stderr, "Greska: neuspesno otvaranje datoteke "
               "ulaz.txt.\n");
14     exit(EXIT_FAILURE);
     }
16
     /* Otvaranje datoteke izlaz.txt za pisanje i provera uspeha. */
18   izlaz = fopen("izlaz.txt", "w");
     if (izlaz == NULL){
20     fprintf(stderr, "Greska: neuspesno otvaranje datoteke "
               "izlaz.txt. \n");
22     exit(EXIT_FAILURE);
     }
24
     /* Citanje karaktera iz ulazne datoteke. */
26   while ((c = fgetc(ulaz)) != EOF){
       /* Upisivanje procitanog karaktera u izlaznu datoteku. */
28     fputc(c, izlaz);

30     /* Preskakanje naredna dva karaktera. */
       fgetc(ulaz);
32     fgetc(ulaz);

34     /* Ovakvo resenje ce raditi i u slucaju kada broj karaktera u
          datoteci nije deljiv sa 3 jer kada se dodje do kraja
36        datoteke svaki sledeci poziv funkcije fgetc vraca EOF. */
     }
38
     /* Zatvaranje otvorenih datoteka. */
40   fclose(izlaz);
     fclose(ulaz);
42
```

```
  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.3

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz, *izlaz;
  char c;

  /* Otvaranje datoteke podaci.txt za citanje i provera uspeha. */
  ulaz = fopen("podaci.txt", "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje datoteke podaci.txt.");

  /* Otvaranje datoteke sifra.txt za pisanje i provera uspeha. */
  izlaz = fopen("sifra.txt", "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje datoteke sifra.txt.");

  /* Citanje karaktera iz ulazne datoteke. */
  while ((c = fgetc(ulaz)) != EOF) {
    /* Sifrovanje procitanog karaktera na trazeni nacin. */
    if (islower(c)) {
      c = toupper(c);
      if (c == 'A')
        c = 'Z';
      else
        c = c - 1;
    } else if (isupper(c)) {
      c = tolower(c);
      if (c == 'a')
        c = 'z';
      else
        c = c - 1;
    }

    /* Upisivanje izmenjenog karaktera u izlaznu datoteku. */
    fputc(c, izlaz);
  }
```

```
47
    /* Zatvaranje datoteka. */
49  fclose(ulaz);
    fclose(izlaz);
51
    exit(EXIT_SUCCESS);
53 }
```

Rešenje 3.3.4

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <ctype.h>

5  #define MAKS_IME 21

7  /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
      izlaz za greske i prekida izvrsavanje programa. */
9  void greska(char *poruka) {
     fprintf(stderr, "%s\n", poruka);
11   exit(EXIT_FAILURE);
   }
13
   int main() {
15   /* Deklaracije potrebnih promenljivih. */
     FILE *ulaz, *izlaz;
17   char c;
     char ime_datoteke1[MAKS_IME], ime_datoteke2[MAKS_IME];
19
     /* Ucitavanje imena datoteka. */
21   printf("Unesite ime prve datoteke: ");
     scanf("%s", ime_datoteke1);
23   printf("Unesite ime druge datoteke: ");
     scanf("%s", ime_datoteke2);
25
     /* Otvaranje prve datoteke za citanje i provera uspeha. */
27   ulaz = fopen(ime_datoteke1, "r");
     if (ulaz == NULL)
29     greska("Greska: neuspesno otvaranje prve datoteke.");

31   /* Otvaranje druge datoteke za pisanje i provera uspeha. */
     izlaz = fopen(ime_datoteke2, "w");
33   if (izlaz == NULL)
       greska("Greska: neuspesno otvaranje druge datoteke.");
35
     /* Iz datoteke se cita karakter po karakter i za svaku procitanu
37      cifru u izlaznu datoteku se upisuje 0, za svako slovo 1, a za
        ostale karaktere 2. */
39   while ((c = fgetc(ulaz)) != EOF) {
       if (isdigit(c))
41       fprintf(izlaz, "0");
```

```
    else if (isalpha(c))
      fprintf(izlaz, "1");
    else
      fprintf(izlaz, "2");
  }

  /* Zatvaranje datoteka. */
  fclose(ulaz);
  fclose(izlaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.5 Pogledajte zadatke 3.3.3 i 3.3.4.

Rešenje 3.3.6 Pogledajte zadatke 3.3.3 i 3.3.4.

Rešenje 3.3.7

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAKS_IME 21
#define BROJ_CIFARA 10

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz;
  char c;
  char ime_datoteke[MAKS_IME];
  int brojaci[BROJ_CIFARA];
  int i, maks;

  /* Ucitavanje imena ulazne datoteke. */
  printf("Unesite ime datoteke: ");
  scanf("%s", ime_datoteke);

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(ime_datoteke, "r");
  if (ulaz == NULL) {
    fprintf(stderr, "Greska: neuspesno otvaranje datoteke.\n");
    exit(EXIT_FAILURE);
  }

  /* Brojaci za cifre se inicijalizuju na nule. Indeks niza brojaci
     oznacava cifru (brojaci[0] se koristi za prebrojavanje cifre
     0, brojaci[1] za 1, ..., brojaci[9] za cifru 9). */
  for (i = 0; i < BROJ_CIFARA; i++)
    brojaci[i] = 0;
```

```
33   /* Citanje karaktera i uvecavanje odgovarajucih brojaca. */
     while ((c = fgetc(ulaz)) != EOF) {
35     if (isdigit(c))
         brojaci[c - '0']++;
37   }

39   /* Pronalazak cifre koja se najvise puta pojavljuje u datoteci. */
     maks = brojaci[0];
41   for (i = 1; i < BROJ_CIFARA; i++)
       if (brojaci[i] > maks)
43       maks = brojaci[i];

45   /* Ispis rezultata. */
     if(maks == 0)
47     printf("Datoteka ne sadrzi cifre.\n");
     else {
49     printf("Najcesce cifre: ");
       for (i = 0; i < BROJ_CIFARA; i++)
51       if (brojaci[i] == maks)
           printf("%d ", i);
53     printf("\n");
     }
55
     /* Zatvaranje datoteke. */
57   fclose(ulaz);

59   exit(EXIT_SUCCESS);
   }
```

Rešenje 3.3.8

```
   #include <stdio.h>
2  #include <stdlib.h>

4  /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
      izlaz za greske i prekida izvrsavanje programa. */
6  void greska(char *poruka) {
     fprintf(stderr, "%s\n", poruka);
8    exit(EXIT_FAILURE);
   }
10
   int main(int argc, char **argv) {
12   /* Deklaracije potrebnih promenljivih. */
     FILE *ulaz;
14   char c;
     int broj_zagrada = 0, nisu_uparene = 0;
16
     /* Provera broja argumenata komandne linije. */
18   if (argc != 2)
       greska("Greska: neispravan poziv.");
20
```

```
  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje datoteke.");

  /* Cita se karakter po karakter i proverava se da li je procitana
     zagrada. Ako se naidje na otvorenu zagradu, brojac se uvecava.
     Ako se naidje na zatvorenu zagradu, brojac se smanjuje. Zagrade
     su ispravno uparene ukoliko je ovaj brojac na kraju 0. Dodatno,
     ukoliko brojac u bilo kom trenutku postane negativan, to znaci
     da je zatvorena zagrada procitana pre otvorene, tako da ni u
     tom slucaju zagrade nisu uparene. */
  while ((c = fgetc(ulaz)) != EOF) {
    if (c == '(')
      broj_zagrada++;
    else if (c == ')')
      broj_zagrada--;

    if (broj_zagrada < 0) {
      nisu_uparene = 1;
      break;
    }
  }

  /* Ispis rezultata. */
  if (broj_zagrada != 0 || nisu_uparene)
    printf("Zagrade nisu uparene.\n");
  else
    printf("Zagrade jesu uparene.\n");

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.9

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

#define MAKS_ELEMENATA 1000

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

```

```
/* Funkcija ucitava karaktere iz datoteke i smesta ih u niz s. */
int ucitaj_karaktere(char s[], FILE *f) {
  char c;
  int n = 0;

  /* Citanje karaktera do kraja datoteke ili dok se ne ucita
     MAKS_ELEMENATA ili dok se ne dodje do karaktera koji nije ni
     slovo ni cifra. */
  while((c = fgetc(f)) != EOF && n < MAKS_ELEMENATA) {
    if(isalpha(c) || isdigit(c))
      s[n] = c;
    else
      break;

    n++;
  }

  return n;
}

/* Funkcija racuna koliko slova i koliko cifara se nalazi u nizu
   s.*/
void prebroj(char s[], int n, int *broj_slova, int *broj_cifara) {
  int i;

  /* Inicijalizacija brojaca. */
  *broj_slova = *broj_cifara = 0;

  /* Prebrojavanje slova i cifara. */
  for(i=0; i<n; i++)
    if(isalpha(s[i]))
      (*broj_slova)++;
    else
      (*broj_cifara)++;
}

int main(int argc, char* argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  FILE* ulaz;
  char karakteri[MAKS_ELEMENATA];
  int broj_elemenata, broj_slova, broj_cifara;

  /* Provera broja argumenata komandne linije. */
  if (argc != 2)
    greska("Greska: neispravan poziv.");

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje datoteke.");

  /* Ucitavanje karaktera datoteke. */
```

```
  broj_elemenata = ucitaj_karaktere(karakteri, ulaz);

  /* Racunanje i ispis rezultata. */
  prebroj(karakteri, broj_elemenata, &broj_slova, &broj_cifara);
  printf("Broj slova: %d\n", broj_slova);
  printf("Broj cifara: %d\n", broj_cifara);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.10

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAKS_NISKA 21

/* Funkcija rotira nisku s duzine n za jedno mesto u levo. */
void rotiraj_za_1(char *s, int n) {
  int i;
  char c = s[0];

  for (i = 0; i < n - 1; i++)
    s[i] = s[i + 1];

  s[n - 1] = c;
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char s[MAKS_NISKA];
  int n, i;
  FILE *izlaz;

  /* Otvaranje datoteke rotacije.txt za pisanje i provera uspeha. */
  izlaz = fopen("rotacije.txt", "w");
  if (izlaz == NULL) {
    fprintf(stderr, "Greska: neuspesno otvaranje datoteke.");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje reci koju je potrebno rotirati. */
  printf("Unesite rec: ");
  scanf("%s", s);

  /* Racunanje duzine reci. */
  n = strlen(s);
```

```
38    /* U petlji se uneta rec rotira za 1 i upisuje u datoteku.
         Postupak se ponavlja n puta. */
40    for (i = 0; i < n; i++) {
        fprintf(izlaz, "%s\n", s);
42      rotiraj_za_1(s, n);
      }
44
      /* Zatvaranje datoteke. */
46    fclose(izlaz);

48    exit(EXIT_SUCCESS);
  }
```

Rešenje 3.3.11

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>

5  #define MAKS_REC 101
   #define MAKS_IME 21
7
   /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
9     izlaz za greske i prekida izvrsavanje programa. */
   void greska(char *poruka) {
11   fprintf(stderr, "%s\n", poruka);
     exit(EXIT_FAILURE);
13 }

15 /* Funkcija u nisku rezultat smesta nisku rec rotiranu za k mesta u
      desno. */
17 void rotiraj(char *rec, int k, char *rezultat) {
     int i, n;
19
     /* Racunanje duzine reci. */
21   n = strlen(rec);

23   /* Ako je duzina reci npr. 5, a k ima vrednost 13, onda je
        zapravo potrebno izvrsiti rotaciju za 3 mesta (nema potrebe da
25      se vrte dva cela kruga pre toga). */
     k = k % n;
27
     /* Karakteri koji se u pocetnoj reci nalaze na pozicijama od 0 do
29      k-1, u rezultujucoj reci treba da budu na pozicijama od n-k do
        n-1. */
31   for (i = 0; i < k; i++)
       rezultat[n - k + i] = rec[i];
33
     /* Slicno, karakteri koji se u pocetnoj reci nalaze na pozicijama
35      od k do n-1, u rezultujucoj reci treba da budu na pozicijama od
```

```
    0 do n-k-1. */
  for (i = k; i < n; i++)
    rezultat[i - k] = rec[i];

  /* Na kraj rezultujuce niske se upisuje terminirajuca nula. */
  rezultat[n] = '\0';
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz, *izlaz;
  char ime_datoteke[MAKS_IME];
  char rec[MAKS_REC], rezultat[MAKS_REC];
  int k;

  /* Ucitavanje imena ulazne datoteke. */
  printf("Unesite ime datoteke: ");
  scanf("%s", ime_datoteke);

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(ime_datoteke, "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Otvaranje datoteke rotirano.txt za citanje i provera uspeha. */
  izlaz = fopen("rotirano.txt", "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje izlazne datoteke.");

  /* Ucitavanje broja k. */
  printf("Unesite broj k: ");
  scanf("%d", &k);
  if (k < 0)
    greska("Greska: neispravan unos broja k.");

  /* Citanje reci iz ulazne datoteke sve do kraja datoteke. */
  while (fscanf(ulaz, "%s", rec) != EOF) {
    /* Rotiranje procitane reci i upisivanje u izlaznu datoteku. */
    rotiraj(rec, k, rezultat);
    fprintf(izlaz, "%s ", rezultat);
  }

  /* Zatvaranje datoteka. */
  fclose(ulaz);
  fclose(izlaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.12 Pogledajte zadatke 3.3.11 i 2.1.18.

Rešenje 3.3.13

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_BROJ_RECI 256
#define MAKS_DUZINA_RECI 51
#define MAKS_IME 21

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  char ime_datoteke[MAKS_IME];
  char niz_reci[MAKS_BROJ_RECI][MAKS_DUZINA_RECI];
  FILE *ulaz, *izlaz;
  int n, i, k, indikator;

  /* Ucitavanje imena ulazne datoteke. */
  printf("Unesite ime datoteke: ");
  scanf("%s", ime_datoteke);

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(ime_datoteke, "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Iz datoteke se ucitava broj reci. */
  fscanf(ulaz, "%d", &n);
  if (n < 0 || n > MAKS_BROJ_RECI)
    greska("Greska: neispravna vrednost broja reci.");

  /* Ucitavanje reci u niz. */
  for (i = 0; i < n; i++)
    fscanf(ulaz, "%s", niz_reci[i]);

  /* Otvaranje izlazne datoteke za pisanje i provera uspeha. */
  izlaz = fopen("bez_duplikata.txt", "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje izlazne datoteke.");

  /* U izlaznu datoteku se upisuju reci, izostavljajuci duplikate. */
  for (i = 0; i < n; i++) {
    /* Za rec na poziciji i se proverava da li se ona nalazi negde
       na pozicijama od 0 do i. Ukoliko se nalazi, to znaci da je
       vec upisana u datoteku i da je treba preskociti. U tom
```

```
51          slucaju vrednost promenljive indikator ce biti postavljena
            na 1. */
53      indikator = 0;
        for (k = 0; k < i; k++)
55        if (strcmp(niz_reci[k], niz_reci[i]) == 0) {
            indikator = 1;
57          break;
          }
59
        /* Ako indikator ima vrednost 0, znaci da je u pitanju prvo
61         pojavljivanje reci i da je treba upisati u izlaznu
           datoteku. */
63      if (!indikator)
          fprintf(izlaz, "%s\n", niz_reci[i]);
65    }

67    /* Zatvaranje datoteka. */
      fclose(ulaz);
69    fclose(izlaz);

71    exit(EXIT_SUCCESS);
  }
```

Rešenje 3.3.14

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <math.h>

5 /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
     izlaz za greske i prekida izvrsavanje programa. */
7 void greska(char *poruka) {
    fprintf(stderr, "%s\n", poruka);
9   exit(EXIT_FAILURE);
  }
11
  /* Funkcija racuna broj cifara broja x. */
13 int broj_cifara(int x) {
    int brojac = 0;
15
    do {
17    brojac++;
      x /= 10;
19  } while (x);

21  return brojac;
  }
23
  /* Funkcija broji koliko ima k-tocifrenih brojeva u datoteci f. */
25 int prebrojavanje(FILE *f, int k) {
    int n, broj, i, brojac;
```

```
27
   /* Ucitavanje broja brojeva u datoteci. */
29 fscanf(f, "%d", &n);
   if (n <= 0)
31   greska("Greska: neispravna vrednost broja n.");

33 /* Cita se broj po broj i za svaki procitani broj se racuna broj
      cifara. Ukoliko je on jednak k, uvecava se odgovarajuci
35    brojac. */
   brojac = 0;
37 for (i = 0; i < n; i++) {
     fscanf(f, "%d", &broj);
39   if (broj_cifara(broj) == k)
       brojac++;
41 }

43 /* Povratna vrednost funkcije je broj k-tocifrenih brojeva. */
   return brojac;
45 }

47 int main(int argc, char *argv[]) {
   /* Deklaracije potrebnih promenljivih. */
49 int k;
   FILE *ulaz;
51
   /* Provera broja argumenata komandne linije. */
53 if (argc != 3)
     greska("Greska: neispravan poziv.");
55
   /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
57 ulaz = fopen(argv[1], "r");
   if (ulaz == NULL)
59   greska("Greska: neuspesno otvaranje ulazne datoteke.");

61 /* Citanje broja k i provera ispravnosti. */
   k = atoi(argv[2]);
63 if (k <= 0)
     greska("Greska: neispravna vrednost broja k.");
65
   /* Ispis rezultata. */
67 printf("Broj %d-cifrenih brojeva: %d\n", k,
          prebrojavanje(ulaz, k));
69
   /* Zatvaranje datoteke. */
71 fclose(ulaz);

73 exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.15

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz;
  float broj, najveci_broj;

  /* Otvaranje datoteke brojevi.txt za citanje i provera uspeha. */
  ulaz = fopen("brojevi.txt", "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Promenljiva u koju se smesta najveci broj se inicijalizuje na
     prvi broj iz datoteke. Ukoliko se pri prvom citanju dodje do
     kraja datoteke, ispisuje se odgovarajuca poruka. */
  if (fscanf(ulaz, "%f", &najveci_broj) == EOF)
    greska("Greska: datoteka je prazna.");

  /* Iz datoteke se cita broj po broj, sve dok se ne dodje do kraja
     datoteke i trazi se najveci procitani broj. */
  while (fscanf(ulaz, "%f", &broj) != EOF)
    if (broj > najveci_broj)
      najveci_broj = broj;

  /* Ispis rezultata. */
  printf("Najveci broj je: %g\n", najveci_broj);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.16

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_DIM 50

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
```

```
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz;
  float a[MAKS_DIM][MAKS_DIM];
  int i, j, n, m, k, l, suma = 0;

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen("matrica.txt", "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Ucitavanje dimenzija matrice i provera ispravnosti. */
  fscanf(ulaz, "%d%d", &n, &m);
  if (n <= 0 || n > MAKS_DIM || m <= 0 || m > MAKS_DIM)
    greska("Greska: neispravne dimenzije matrice.");

  /* Ucitavanje elemenata matrice. */
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      fscanf(ulaz, "%f", &a[i][j]);

  /* Za svaku poziciju (i,j) vrsi se provera trazenog uslova. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
      /* Za poziciju (i,j) racuna se suma suseda. Ona se moze
         dobiti kao suma podmatrice 3*3 ciji je gornji levi ugao
         (i-1, j-1), a donji desni (i+1, j+1). Pri racunanju ove
         sume treba voditi racuna da se ne izadje iz granica
         originalne matrice. */
      suma = 0;
      for (k = i - 1; k <= i + 1; k++) {
        for (l = j - 1; l <= j + 1; l++) {
          /* Ako se nije izaslo iz granica originalne matrice,
             vrednost a[k][l] se dodaje na sumu. */
          if (k >= 0 && k < n && l >= 0 && l < m)
            suma += a[k][l];
        }
      }

      /* Kako suma ukljucuje i centralni element (i,j), njega je
         potrebno oduzeti jer je potrebno sumirati samo njegove
         susede. */
      suma -= a[i][j];

      /* Ako je suma suseda elementa a[i][j] jednaka njegovoj
         vrednosti, odgovarajuce pozicije i vrednost elementa se
```

```
          ispisuju na standardni izlaz. */
60      if (a[i][j] == suma)
          printf("(%d, %d, %g)\n", i, j, a[i][j]);
62    }
  }
64
  /* Zatvaranje datoteke. */
66  fclose(ulaz);

68  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.17 Pogledajte zadatak 3.3.16.

Rešenje 3.3.18

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <math.h>

5 #define MAKS_TACAKA 50

7 /* Struktura koja opisuje tacku. */
  typedef struct {
9   int x, y;
  } Tacka;
11
  /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
13    izlaz za greske i prekida izvrsavanje programa. */
  void greska(char *poruka) {
15  fprintf(stderr, "%s\n", poruka);
    exit(EXIT_FAILURE);
17 }

19 /* Funkcija racuna rastojanje tacke od koordinatnog pocetka. */
  double rastojanje_od_koordinatnog_pocetka(const Tacka *a) {
21  return sqrt(pow(a->x, 2) + pow(a->y, 2));
  }
23
  int main() {
25  /* Deklaracije potrebnih promenljivih. */
    FILE *ulaz, *izlaz;
27  int n, i;
    Tacka tacka, maks_tacka;
29  double rastojanje, maks_rastojanje = -1;

31  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
    ulaz = fopen("tacke.txt", "r");
33  if (ulaz == NULL)
      greska("Greska: neuspesno otvaranje ulazne datoteke.");
35
```

```
  /* Otvaranje izlazne datoteke za pisanje i provera uspeha. */
  izlaz = fopen("rastojanja.txt", "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje izlazne datoteke.");

  /* Ucitavanje broja tacaka i provera ispravnosti. */
  fscanf(ulaz, "%d", &n);
  if (n < 0 || n > MAKS_TACAKA)
    greska("Greska: neispravan broj tacaka.");

  /* Citanje tacaka iz datoteke. */
  for (i = 0; i < n; i++) {
    fscanf(ulaz, "%d%d", &tacka.x, &tacka.y);

    /* Racunanje rastojanja tacke t od koordinatnog pocetka. */
    rastojanje = rastojanje_od_koordinatnog_pocetka(&tacka);

    /* Upisivanje izracunatog rastojanja u datoteku. */
    fprintf(izlaz, "%.2lf\n", rastojanje);

    /* Azuriranje maksimalnog rastojanja i odgovarajuce tacke. */
    if (rastojanje > maks_rastojanje) {
      maks_rastojanje = rastojanje;
      maks_tacka = tacka;
    }
  }

  /* Ispis rezultata. */
  printf("Najudaljenija tacka: (%d, %d)\n", maks_tacka.x, maks_tacka.
    y);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.19 Pogledajte zadatak 3.3.18.

Rešenje 3.3.20

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_IME 5

/* Struktura koja opisuje pravougaonik. */
typedef struct {
  unsigned int a, b;
  char ime[MAKS_IME];
```

```
} Pravougaonik;

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  unsigned int maksimalna_povrsina = 0;
  Pravougaonik p;

  /* Provera broja argumenata komandne linije. */
  if (argc != 2)
    greska("Greska: neispravan poziv.");

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  FILE *ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Citanje podataka o pravougaonicima. */
  while (fscanf(ulaz, "%u%u%s", &p.a, &p.b, p.ime) == 3) {
    /* Provera ispravnosti duzina stranica. */
    if (p.a == 0 || p.b == 0)
      greska("Greska: duzina stranice ne moze biti 0.");

    /* U slucaju da je ucitan kvardat, njegovo ime se ispisuje na
       standardni izlaz. */
    if (p.a == p.b)
      printf("%s ", p.ime);
    else { /* Ako je u pitanju pravougaonik, njegova povrsina
              se poredi sa maksimalnom. */
      if (p.a * p.b > maksimalna_povrsina)
        maksimalna_povrsina = p.a * p.b;
    }
  }

  /* Ukoliko je bilo ucitanih pravougaonika, ispisuje se povrsina
     najveceg. */
  if (maksimalna_povrsina != 0)
    printf("%u\n", maksimalna_povrsina);
  else
    printf("\n");

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.21

```
#include <stdio.h>
#include <stdlib.h>

#define MAKS_STUDENATA 100

/* Struktura koja opisuje studenta. */
typedef struct {
  char korisnicko_ime[8];
  float prosek;
} Student;

int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz;
  Student studenti[MAKS_STUDENATA];
  int ocena1, ocena2, ocena3, ocena4, ocena5, zbir_ocena;
  int i = 0, n;
  float maksimalni_prosek = 0;

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen("studenti.txt", "r");
  if (ulaz == NULL) {
    fprintf(stderr, "Greska: neuspesno otvaranje "
            "ulazne datoteke.\n");
    exit(EXIT_FAILURE);
  }

  /* Citanje podataka o studentima sve dok se ne dodje do kraja
     datoteke. */
  while (fscanf(ulaz, "%s%d%d%d%d%d", studenti[i].korisnicko_ime,
          &ocena1, &ocena2, &ocena3, &ocena4, &ocena5) != EOF) {
    /* Racunanje proseka trenutnog studenta. */
    zbir_ocena = ocena1 + ocena2 + ocena3 + ocena4 + ocena5;
    studenti[i].prosek = zbir_ocena / 5.0;

    /* Azuriranje maksimalnog proseka. */
    if (studenti[i].prosek > maksimalni_prosek)
      maksimalni_prosek = studenti[i].prosek;

    /* Prelazak na sledeceg studenta. */
    i++;
  }

  /* Promenljiva n cuva ukupan broj studenata. */
  n = i;

  /* Ispis svih studenata sa maksimalnim prosekom. */
  printf("Studenti sa najvecim prosekom:\n");
  for (i = 0; i < n; i++)
    if (studenti[i].prosek == maksimalni_prosek)
```

```
      printf("Korisnicko ime: %s\nProsek ocena: %.2f\n\n",
52            studenti[i].korisnicko_ime, studenti[i].prosek);

54  /* Zatvaranje datoteke. */
    fclose(ulaz);
56
    exit(EXIT_SUCCESS);
58 }
```

Rešenje 3.3.22

```
#include <stdio.h>
2 #include <stdlib.h>
  #include <string.h>
4
  #define MAKS_PUNO_IME 101
6 #define MAKS_OCENA 10

8 /* Struktura koja opisuje studenta. */
  typedef struct {
10   char puno_ime[MAKS_PUNO_IME];
    int ocene[MAKS_OCENA];
12   int broj_ocena;
    float prosek;
14 } Student;

16 /* Funkcija ispisuje prosledjenu poruku o gresci na standardni
     izlaz za greske i prekida izvrsavanje programa. */
18 void greska(char *poruka) {
    fprintf(stderr, "%s\n", poruka);
20   exit(EXIT_FAILURE);
  }
22
  int main(int argc, char **argv) {
24   /* Deklaracije potrebnih promenljivih. */
    FILE *ulaz;
26   char ime[MAKS_PUNO_IME], prezime[MAKS_PUNO_IME];
    int i = 0, j, ocena, suma_ocena;
28   Student student, maks_student;
    float maks_prosek;
30
    /* Provera broja argumenata komandne linije. */
32   if (argc != 2)
      greska("Greska: neispravan poziv.");
34
    /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
36   ulaz = fopen(argv[1], "r");
    if (ulaz == NULL)
38     greska("Greska: neuspesno otvaranje ulazne datoteke.");

40   /* Iz datoteke se ucitavaju podaci o studentima sve dok se ne
```

```
     dodje do kraja datoteke. */
  while (fscanf(ulaz, "%s%s", ime, prezime) != EOF) {
    /* Od imena i prezimena se formira puno ime. */
    strcpy(student.puno_ime, ime);
    strcat(student.puno_ime, " ");
    strcat(student.puno_ime, prezime);

    /* Ucitavanje ocena sve dok se ne ucita broj 0. */
    j = 0;
    suma_ocena = 0;
    while (1) {
      fscanf(ulaz, "%d", &ocena);
      if (ocena == 0)
        break;

      student.ocene[j] = ocena;
      suma_ocena += ocena;
      j++;
    }

    /* Racunanje proseka ocena. */
    student.broj_ocena = j;
    student.prosek = (float) suma_ocena / j;

    /* Ukoliko je u pitanju student ciji je prosek veci od trenutno
       najveceg proseka, pamte se njegovi podaci i azurira se
       vrednost najveceg proseka. */
    if (student.prosek > maks_prosek) {
      maks_prosek = student.prosek;
      maks_student = student;
    }
  }

  /* Ispis podataka o studentu sa najvecim prosekom. */
  printf("%s ", maks_student.puno_ime);
  for (i = 0; i < maks_student.broj_ocena; i++)
    printf("%d ", maks_student.ocene[i]);
  printf("%.2f\n", maks_student.prosek);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.23

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
```

```
5
#define MAKS_RAZLOMAKA 100
7
/* Struktura koja opisuje razlomak. */
9 typedef struct {
    int brojilac;
11  int imenilac;
} Razlomak;
13
/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
15     izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
17  fprintf(stderr, "%s\n", poruka);
    exit(EXIT_FAILURE);
19 }

21 /* Funkcija ucitava razlomke u niz razlomaka i kao povratnu
     vrednost vraca broj ucitanih razlomaka. */
23 int ucitaj_razlomke(Razlomak niz[], FILE *f) {
    int i, n;
25  fscanf(f, "%d", &n);

27  for (i = 0; i < n; i++) {
      fscanf(f, "%d %d", &niz[i].brojilac, &niz[i].imenilac);
29    if (niz[i].imenilac == 0)
        greska("Greska: Imenilac ne moze biti 0.");
31  }
    return n;
33 }

35 /* Funkcija racuna razlomak reciprocan razlomku r. */
Razlomak reciprocni(const Razlomak *r) {
37  if (r->brojilac == 0)
      greska("Greska: nije moguce izracunati reciprocni razlomak.");
39
    Razlomak r2;
41  r2.imenilac = r->brojilac;
    r2.brojilac = r->imenilac;
43  return r2;
}
45
/* Funkcija racuna brojevnu vrednost razlomka r. */
47 float vrednost(const Razlomak *r) {
    return 1.0 * r->brojilac / r->imenilac;
49 }

51 int main(int argc, char *argv[]) {
    /* Deklaracije potrebnih promenljivih. */
53  FILE *ulaz, *izlaz;
    int i, n, opcija_x = 0, opcija_y = 0;
55  Razlomak razlomci[MAKS_RAZLOMAKA];
    Razlomak r;
```

```
57
   /* Provera broja argumenata komandne linije. */
59 if (argc != 4)
     greska("Greska: neispravan poziv.");
61
   /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
63 ulaz = fopen(argv[1], "r");
   if (ulaz == NULL)
65   greska("Greska: neuspesno otvaranje ulazne datoteke.");

67 /* Otvaranje izlazne datoteke za pisanje i provera uspeha. */
   izlaz = fopen(argv[2], "w");
69 if (izlaz == NULL)
     greska("Greska: neuspesno otvaranje izlazne datoteke.");
71
   /* Ucitavanje zadate opcije i postavljanje vrednosti
73    odgovarajuceg indikatora. */
   if (strcmp(argv[3], "-x") == 0)
75   opcija_x = 1;
   else if (strcmp(argv[3], "-y") == 0)
77   opcija_y = 1;
   else if (strcmp(argv[3], "-xy") == 0
79            || strcmp(argv[3], "-yx") == 0)
     opcija_x = opcija_y = 1;
81 else
     greska("Greska: neispravna opcija.");
83
   /* Ucitavanje podataka o razlomcima. */
85 n = ucitaj_razlomke(razlomci, ulaz);

87 /* Prolazak kroz niz razlomaka. */
   for (i = 0; i < n; i++) {
89   /* Racunanje reciprocnog razlomka. */
     r = reciprocni(&razlomci[i]);
91
     /* Ispis rezultata u zavisnosti od navedenih opcija. */
93   if (opcija_x)
       fprintf(izlaz, "%d/%d ", r.brojilac, r.imenilac);
95   if (opcija_y)
       fprintf(izlaz, "%f ", vrednost(&r));
97   fprintf(izlaz, "\n");
   }
99
   /* Zatvaranje datoteka. */
101 fclose(ulaz);
   fclose(izlaz);
103
   exit(EXIT_SUCCESS);
105 }
```

Rešenje 3.3.24

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_IME 31
#define MAKS_AUTOMOBILA 100

/* Struktura koja opisuje automobil. */
typedef struct {
  char marka[MAKS_IME];
  char model[MAKS_IME];
  float cena;
} Automobil;

/*
   Struktura Info sadrzi naziv marke automobila, prosek cena za tu
   marku i broj automobila te marke */
typedef struct {
  char marka[MAKS_IME];
  float prosecna_cena;
  int n;
} Info;

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

/* Funkcija ucitava informacije o automobilima iz fajla i smesta ih
   u niz struktura. Kao povratnu vrednost funkcija vraca broj
   ucitanih automobila. */
int ucitaj(FILE *f, Automobil a[]) {
  int i, n;

  fscanf(f, "%d", &n);
  if (n <= 0 || n > MAKS_AUTOMOBILA)
    greska("Greska: neispravan broj automobila.");

  for (i = 0; i < n; i++)
    fscanf(f, "%s %s %f", a[i].marka, a[i].model, &a[i].cena);

  return n;
}

/* Funkcija proverava da li se u nizu sa informacijama o markama
   nalazi prosledjena marka. Ukoliko se nalazi, vraca odgovarajucu
   poziciju, a u suprotnom vraca -1. */
int sadrzi(Info info[], int n, char marka[]) {
```

```
  int i;
  for (i = 0; i < n; i++)
    if (strcmp(info[i].marka, marka) == 0)
      return i;

  return -1;
}

/* Funkcija popunjava niz sa informacijama o markama na osnovu
   podataka datih u nizu automobila. */
void izracunaj_proseke(Automobil a[], int automobili_n,
                       Info info[], int *n) {
  int i, pozicija, j = 0;
  for (i = 0; i < automobili_n; i++) {
    pozicija = sadrzi(info, j, a[i].marka);
    if (pozicija == -1) {
      strcpy(info[j].marka, a[i].marka);
      info[j].prosecna_cena = a[i].cena;
      info[j].n = 1;
      j++;
    } else {
      info[pozicija].prosecna_cena += a[i].cena;
      info[pozicija].n += 1;
    }
  }

  for (i = 0; i < j; i++)
    info[i].prosecna_cena /= info[i].n;

  *n = j;
}

/* Funkcija ispisuje informacije o prosecnim cenama za svaku
   marku. */
void ispisi_informacije(Info info[], int n) {
  int i;
  printf("Informacije o prosecnoj ceni po markama:\n");
  for (i = 0; i < n; i++)
    printf("%s %.2f\n", info[i].marka, info[i].prosecna_cena);
}

/* Funkcija ispisuje podatke o automobilima cija je cena manja ili
   jednaka budzetu kojim korisnik raspolaze. */
void ispisi_kandidate(Automobil a[], int automobili_n,
                      float budzet) {
  int i;
  printf("Kola u Vasem cenovnom rangu:\n");
  for (i = 0; i < automobili_n; i++)
    if (a[i].cena < budzet)
      printf("%s %s %g\n", a[i].marka, a[i].model, a[i].cena);
}
```

```
int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  Automobil automobili[MAKS_AUTOMOBILA];
  FILE *ulaz;
  char ime_datoteke[MAKS_IME];
  float budzet;
  Info info[MAKS_AUTOMOBILA];
  int automobili_n, info_n;

  /* Provera broja argumenata komandne linije. */
  if (argc != 2)
    greska("Greska: neispravan poziv.");

  /* Ucitavanje budzeta. */
  budzet = atof(argv[1]);

  /* Ucitavanje naziva datoteke. */
  printf("Unesite naziv datoteke: ");
  scanf("%s", ime_datoteke);

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(ime_datoteke, "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Ucitavanje podataka o automobilima. */
  automobili_n = ucitaj(ulaz, automobili);

  /* Racunanje proseka za svaku marku. */
  izracunaj_proseke(automobili, automobili_n, info, &info_n);

  /* Ispis podataka za sve marke automobila. */
  ispisi_informacije(info, info_n);

  /* Ispis podataka o automobilima cija je cena manja ili
     jednaka granici koju je korisnik uneo. */
  ispisi_kandidate(automobili, automobili_n, budzet);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.25

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAKS_LINIJA 81
```

```
6
/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
8    izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
10  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
12 }

14 int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
16  FILE *ulaz;
  char linija[MAKS_LINIJA];
18  int k;

20  /* Provera broja argumenata komandne linije. */
  if (argc != 3)
22    greska("Greska: neispravan poziv.");

24  /* Otvaranje datoteke cije se ime zadaje kao prvi argument
     komandne linije i provera uspeha. */
26  ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
28    greska("Greska: neuspesno otvaranje ulazne datoteke.");

30  /* Racunanje vrednosti drugog argumenta komandne linije. */
  k = atoi(argv[2]);
32
  /* Funkcija fgets cita jednu liniju iz datoteke. Njeni
34     argumenti su:
     1. Niska u koju ce biti smestena procitana linija
36     2. Maksimalna duzina linije
     3. Datoteka iz koje se cita.
38     Kada dodje do kraja datoteke, kao povratnu vrednost
     funkcija vraca NULL. */
40  while (fgets(linija, MAKS_LINIJA, ulaz) != NULL) {
    /* Ispis svih linija cija je duzina veca od k. */
42    if (strlen(linija) > k)
      printf("%s", linija);
44  }
  printf("\n");
46
  /* Zatvaranje datoteke. */
48  fclose(ulaz);

50  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.26

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_LINIJA 81

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  char linija[MAKS_LINIJA], najduza_linija[MAKS_LINIJA];
  int duzina, maks_duzina;
  FILE *ulaz;

  /* Provera broja argumenata komandne linije. */
  if (argc != 2)
    greska("Greska: neispravan poziv.");

  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Pronalazak najduze linija u datoteci. */
  maks_duzina = 0;
  while (fgets(linija, MAKS_LINIJA, ulaz) != NULL) {
    duzina = strlen(linija);

    if (duzina > maks_duzina ||
        (duzina == maks_duzina &&
         strcmp(linija, najduza_linija) < 0)) {
      strcpy(najduza_linija, linija);
      maks_duzina = duzina;
    }
  }

  /* Ispis najduze linije na standardni izlaz. */
  printf("%s", najduza_linija);

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.27

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAKS_LINIJA 81
#define MAKS_REC 31

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

/* Funkcija broji koliko puta se niska t javlja u okviru niske s. */
int broj_pojavljivanja(char s[], char t[]) {
  int brojac = 0, i;
  int tn = strlen(t);
  int sn = strlen(s);

  /* Funkcija strncmp(s,t,n) poredi prvih n karaktera niski s i t.
     U petlji se vrsi poredjenje niske t sa svim podniskama niske s
     cija je duzina tn.
     Na primer, ako je s = "abcab", a t = "ab", tada je sn = 5,
     a tn = 2.
     Za i = 0, zove se strncmp("abcab", "ab", 2) i na taj nacin se
               porede "ab" i "ab".
     Za i = 1, zove se strncmp("bcab", "ab", 2) i na taj nacin se
               porede "bc" i "ab".
     ...
     Za i = sn - st = 5 - 2 = 3, zove se strncmp("ab", "ab", 2) i
               na taj nacin se porede "ab" i "ab". */
  for (i = 0; i <= sn - tn; i++)
    if (strncmp(s + i, t, tn) == 0)
      brojac++;

  return brojac;
}

int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  char rec[MAKS_REC];
  char linija[MAKS_LINIJA];
  FILE *ulaz, *izlaz;
  int n, brojac;

  /* Provera broja argumenata komandne linije. */
  if (argc != 3)
    greska("Greska: neispravan poziv.");

```

```
  /* Otvaranje ulazne datoteke za citanje i provera uspeha. */
  ulaz = fopen(argv[1], "r");
  if (ulaz == NULL)
    greska("Greska: neuspesno otvaranje ulazne datoteke.");

  /* Otvaranje izlazne datoteke za pisanje i provera uspeha. */
  izlaz = fopen(argv[2], "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje izlazne datoteke.");

  /* Ucitavanje broja n i provera ispravnosti unosa. */
  printf("Unesite broj n: ");
  scanf("%d", &n);
  if (n <= 0)
    greska("Greska: neispravan unos.");

  /* Ucitavanje trazene reci. */
  fscanf(ulaz, "%s", rec);

  /* Iz ulazne datoteke se cita linija po linija i u izlaznu
     datoteku se upisuju sve linije koje trazenu rec sadrze bar n
     puta. */
  while (fgets(linija, MAKS_LINIJA, ulaz) != NULL) {
    brojac = broj_pojavljivanja(linija, rec);
    if (brojac >= n)
      fprintf(izlaz, "%d: %s", brojac, linija);
  }

  /* Zatvaranje datoteka. */
  fclose(ulaz);
  fclose(izlaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.28

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAKS_LINIJA 81
#define MAKS_NISKA 21

/* Funkcija prebrojava koliko linija datoteke ulaz se zavrsava
   niskom s. */
int broj_linija(FILE *ulaz, char *s) {
  char linija[MAKS_LINIJA];
  int brojac = 0, duzina_linije;
  int duzina_s = strlen(s);

```

```
  /* Citanje linija iz datoteke sve do kraja datoteke. */
  while (fgets(linija, MAKS_LINIJA, ulaz) != NULL) {
    /* Racunanje duzine procitane linije. */
    duzina_linije = strlen(linija);

    /* Uklanjanje znaka za novi red sa kraja linije. */
    if (linija[duzina_linije - 1] == '\n') {
      linija[duzina_linije - 1] = '\0';
      duzina_linije--;
    }

    /* Poredjenje kraja linije sa niskom s. Kraj linije se moze
       dobiti tako sto se izvrsi 'pomeranje' u desno do kraja
       linije, a zatim 'pomeranje' u levo onoliko mesta koliko je
       dugacka niska s.
       Na primer, ako je linija "abcdefghijk", a niska s "ab",
       onda se sa linija + duzina_linije vrsi pomeranje na karakter
       iza karaktera 'k' (odnosno null-terminator), a sa
       linija + duzina_linije - duzina_s
       na karakter 'j'. Ukoliko se funkcija strcmp pozove sa
       strcmp(linija + duzina_linije - duzina_s, s),
       vrsice se poredjenje niske "jk" i "ab", sto je i bio cilj. */
    if (strcmp(linija + duzina_linije - duzina_s, s) == 0)
      brojac++;
  }

  return brojac;
}


int main() {
  /* Deklaracije potrebnih promenljivih. */
  FILE *ulaz;
  char s[MAKS_NISKA];

  /* Otvaranje datoteke ulaz.txt za citanje i provera uspeha. */
  ulaz = fopen("ulaz.txt", "r");
  if (ulaz == NULL) {
    fprintf(stderr, "Greska: neuspesno otvaranje ulazne "
            "datoteke.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje niske s. */
  printf("Unesite nisku s: ");
  scanf("%s", s);

  /* Ispis rezultata. */
  printf("Broj linija: %d\n", broj_linija(ulaz, s));

  /* Zatvaranje datoteke. */
  fclose(ulaz);
```

```
68    exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.29

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include <stdlib.h>

#define MAKS_LINIJA 81

/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
   izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  char linija[MAKS_LINIJA];
  FILE *izlaz;
  int ispis_velika_slova = 0, ispis_mala_slova = 0;

  /* Provera broja argumenata komandne linije. */
  if (argc > 2)
    greska("Greska: neispravan poziv.");

  /* Postavljanje vrednosti indikatora za ispis u zavisnosti od
     navedene opcije. */
  if (argc == 1)
    ispis_velika_slova = ispis_mala_slova = 1;
  else {
    /* Funkcija strcasecmp poredi niske ignorisuci razliku izmedju
       malih i velikih slova. */
    if (strcasecmp(argv[1], "-v") == 0)
      ispis_velika_slova = 1;
    else if (strcasecmp(argv[1], "-m") == 0)
      ispis_mala_slova = 1;
    else
      greska("Greska: neispravna opcija.");
  }

  /* Otvaranje datoteke izlaz.txt za pisanje i provera uspeha. */
  izlaz = fopen("izlaz.txt", "w");
  if (izlaz == NULL)
    greska("Greska: neuspesno otvaranje izlazne datoteke.");

  /* Citanje linija sa standardnog ulaza i ispis odgovarajucih
```

```
46      linija u izlaznu datoteku. */
    printf("Unesite recenice: \n");
48    while (fgets(linija, MAKS_LINIJA, stdin) != NULL) {
      if ((ispis_mala_slova && islower(linija[0])) ||
50        (ispis_velika_slova && isupper(linija[0])) ||
          (ispis_mala_slova && ispis_velika_slova))
52        fputs(linija, izlaz);
    }
54
    /* Zatvaranje datoteke. */
56  fclose(izlaz);

58  exit(EXIT_SUCCESS);
}
```

Rešenje 3.3.30

```
#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAKS_LINIJA 201
6
/* Funkcija ispisuje prosledjenu poruku o gresci na standardni
8    izlaz za greske i prekida izvrsavanje programa. */
void greska(char *poruka) {
10  fprintf(stderr, "%s\n", poruka);
  exit(EXIT_FAILURE);
12 }

14 int main(int argc, char **argv) {
  /* Deklaracije potrebnih promenljivih. */
16  int i = 1;
  char *d1, *d2;
18  FILE *ulaz1, *ulaz2;
  char linija1[MAKS_LINIJA], linija2[MAKS_LINIJA];
20
  /* Provera broja argumenata komandne linije. */
22  if (argc != 3)
    greska("Greska: neispravan poziv.");
24
  /* Otvaranje ulaznih datoteka za citanje i provera uspeha. */
26  ulaz1 = fopen(argv[1], "r");
  ulaz2 = fopen(argv[2], "r");
28  if (ulaz1 == NULL || ulaz2 == NULL)
    greska("Greska: neuspesno otvaranje datoteke.");
30
  /* Citanje prve linije iz obe datoteke. */
32  d1 = fgets(linija1, MAKS_LINIJA, ulaz1);
  d2 = fgets(linija2, MAKS_LINIJA, ulaz2);
34
  /* Citanje preostalih linija dok se ne dodje do kraja bar jedne
```

```
36      datoteke. */
    while (d1 != NULL && d2 != NULL) {
38    /* Poredjenje ucitanih linija. */
      if (strcmp(linija1, linija2) != 0)
40      printf("%d ", i);

42    /* Prelazak na sledece linije. */
      d1 = fgets(linija1, MAKS_LINIJA, ulaz1);
44    d2 = fgets(linija2, MAKS_LINIJA, ulaz2);

46    i++;
    }
48
    /* Iz prethodne petlje je moglo da se izadje u 3 slucaja:
50     1. Doslo se do kraja prve datoteke.
       2. Doslo se do kraja druge datoteke.
52     3. Doslo se do kraja obeju datoteka.
       U slucaju da se desio treci slucaj, nijedna od naredne dve
54     petlje se nece izvrsiti. U prvom slucaju ce se izvrsiti samo
       prva petlja, a u drugom slucaju druga. */
56
    /* Ispis preostalih rednih brojeva linija prve datoteke. */
58  while (d1 != NULL) {
      printf("%d ", i);
60    d1 = fgets(linija1, MAKS_LINIJA, ulaz1);
      i++;
62  }

64  /* Ispis preostalih rednih brojeva linija druge datoteke. */
    while (d2 != NULL) {
66    printf("%d ", i);
      d2 = fgets(linija2, MAKS_LINIJA, ulaz2);
68    i++;
    }
70
    /* Zatvaranje datoteka. */
72  fclose(ulaz1);
    fclose(ulaz2);
74
    exit(EXIT_SUCCESS);
76  }
```

Dodatak A

Ispitni rokovi

A.1 Opšta grupa

A.1.1 Praktični deo ispita, januar 2019.

Zadatak A.1.1 Napisati program koji učitava četvorocifrene brojeve do unosa broja 0, a zatim ispisuje one brojeve kojima je cifra desetica najveća cifra u zapisu. Ukoliko nema takvih brojeva među unetima, ispisati broj 0. U slučaju greške, ispisati -1 na standardni izlaz za greške.

Test 1

```
Ulaz:
 9523 -8542 3232 -9999 -1121 1576 0
Izlaz:
 3232 -9999 -1121 1576
```

Test 2

```
Ulaz:
 4596 1234 9631 -120 0
Izlaz:
 4596
Izlaz za greške:
 -1
```

Test 3

```
Ulaz:
 9876 2258 -4579 4689 -5567 6630 1200 5204 0
Izlaz:
 0
```

Zadatak A.1.2 Napisati program koji pomaže korisniku da "šifruje" svoju elektronsku adresu kako ne bi dobijao nepoželjne poruke. "Šifrovanje" adrese se vrši tako što se znak @ zameni sa $[AT]$. Elektronska adresa se učitava kao niska maksimalne dužine 100 karaktera sa standardnog ulaza, a šifrovana adresa

se ispisuje na standardni izlaz. U slučaju da elektronska adresa nije ispravno zadata ispisati -1 na standardni izlaz za greške.

Test 1

```
ULAZ:
  korisnik@gmail.com
IZLAZ:
  korisnik[AT]gmail.com
```

Test 2

```
ULAZ:
  student@matf.bg.ac.rs
IZLAZ:
  student[AT]matf.bg.ac.rs
```

Test 3

```
ULAZ:
  pogresnaadresayahoo.com
IZLAZ ZA GREŠKE:
  -1
```

Zadatak A.1.3 Definisati strukturu *Hemijski_element* koja sadrži naziv elementa (nisku dužine najviše 20 karaktera), oznaku elementa (nisku dužine najviše 2 karaktera) i broj neutrona (ceo broj). Napisati program koji učitava podatke o hemijskim elementima do unosa reči **kraj**, a potom još jedan naziv elementa i na standardni izlaz ispisuje oznaku i broj neutrona tog elementa. Ukoliko element nije pronađen među učitanim podacima, ispisati -1.
NAPOMENA: *Pretpostaviti da neće biti uneto više od 120 elemenata, kao i da su podaci o hemijskim elementima ispravno zadati.*

Test 1

```
ULAZ:
  kalcijum Ca 20
  cink Zn 35
  fosfor P 16
  kraj
  fosfor
IZLAZ:
  P 16
```

Test 2

```
ULAZ:
  nikl Ni 31
  bor B 6
  kripton Kr 48
  natrijum Na 12
  kraj
  hrom
IZLAZ ZA GREŠKE:
  -1
```

Test 3

```
ULAZ:
  litijum Li 4
  ugljenik C 6
  aluminijum Al 14
  srebro Ag 61
  gvozdje Fe 40
  brom Br 45
  kraj
  ugljenik
IZLAZ:
  C 6
```

Zadatak A.1.4 U datoteci *pesme.txt* dat je ceo broj n koji označava broj pesama, a potom i n redova sa podacima o pesmama. U svakom redu naveden je naziv pesme i njen žanr (niske bez belina, dužine najviše 30 karaktera). Napisati program koji učitava podatke iz datoteke, a zatim, u zavisnosti od opcije koja se zadaje kao argument komandne linije, obrađuje podatke na sledeći način:

- ukoliko je zadata opcija **-p**, učitava se sa standardnog ulaza jedan karakter i na standardni izlaz ispisuju svi nazivi pesama koji počinju zadatim karakterom;
- ukoliko je zadata opcija **-z**, učitava se sa standardnog ulaza niska koja predstavlja žanr pesme i na standardni izlaz ispisuju nazivi svih pesama odabranog žanra.

Prilikom odabira pesama za ispis, zanemariti veličinu slova. U slučaju greške, ispisati -1 na standardni izlaz za greške.

Test 1

```
Pokretanje: ./a.out -p
pesme.txt
 7
 BohemianRhapsody rock
 RollingInTheDeep pop
 StairwayToHeaven rock
 BeatIt pop
 SoWhat jazz
 MyFunnyValentine jazz
 Smooth pop
Ulaz:
 S
Izlaz:
 StairwayToHeaven
 SoWhat
 Smooth
```

Test 2

```
Pokretanje: ./a.out -z
pesme.txt
 7
 BohemianRhapsody rock
 RollingInTheDeep pop
 StairwayToHeaven rock
 BeatIt pop
 SoWhat jazz
 MyFunnyValentine jazz
 Smooth pop
Ulaz:
 pop
Izlaz:
 RollingInTheDeep
 BeatIt
 Smooth
```

Test 3

```
Pokretanje: ./a.out -x
Izlaz za greške:
 -1
```

Test 4

```
Pokretanje: ./a.out -p -z
Izlaz za greške:
 -1
```

Test 5

```
Pokretanje: ./a.out
Izlaz za greške:
 -1
```

A.1.2 Praktični deo ispita, februar 2019.

Zadatak A.1.5 Napisati program koji učitava pozitivan četvorocifren broj n, a zatim na standardni izlaz ispisuje zbir onih cifara broja n koje su po vrednosti veće od aritmetičke sredine svih cifara broja n. U slučaju greške, ispisati -1 na standardni izlaz za greške.

Test 1

```
Ulaz:
 1234
Izlaz:
 7
```

Test 2

```
Ulaz:
 6745
Izlaz:
 13
```

Test 3

```
Ulaz:
 100
Izlaz za greške:
 -1
```

Test 4

```
Ulaz:
 -1234
Izlaz za greške:
 -1
```

Zadatak A.1.6 Napisati program koji učitava nisku s parne dužine od najviše 20 karaktera i na standardni izlaz ispisuje nisku koja se dobija nadovezivanjem karaktera prve polovine niske s na drugu polovinu niske s. U slučaju greške, ispisati -1 na standardni izlaz za greške.

Test 1

```
ULAZ:
  Beograde
IZLAZ:
  radeBeog
```

Test 2

```
ULAZ:
  matematika
IZLAZ:
  atikamatem
```

Test 3

```
ULAZ:
  1234
IZLAZ:
  3412
```

Test 4

```
ULAZ:
  abc1234
IZLAZ ZA GREŠKE:
  -1
```

Zadatak A.1.7 Napisati program koji čita sadržaj datoteke *ulaz.txt* i ispisuje na standardni izlaz sve niske datoteke koje predstavljaju cele brojeve. U slučaju greške, ispisati -1 na standardni izlaz za greške.

Test 1

```
POKRETANJE: ./a.out
ULAZ.TXT
  123 ab1 2ab -23
IZLAZ:
  123 -23
```

Test 2

```
POKRETANJE: ./a.out
ULAZ.TXT
  145as 25gf 265 478 65 -96
IZLAZ:
  265 478 65 -96
```

Test 3

```
POKRETANJE: ./a.out
ULAZ.TXT
  Ovde nema brojeva
IZLAZ:
```

Test 4

```
POKRETANJE: ./a.out
ULAZ.TXT NE POSTOJI!
IZLAZ ZA GREŠKE:
  -1
```

Zadatak A.1.8 Napisati program koji sa standardnog ulaza učitava podatke o osvajačima takmičenja. Za svako takmičenje se redom zadaju godina takmičenja (pozitivan ceo broj) i ime osvajača (niska od najviše 30 karaktera bez belina). Program treba da ispiše:

- ako je navedena opcija -y kao prvi argument komandne linije, ime osvajača takmičenja za godinu koja se navodi kao drugi argument
- ako je navedena opcija -w kao prvi argument komandne linije, sve godine u kojima je takmičar čije se ime navodi kao drugi argument komande linije osvajao takmičenje.

U slučaju greške, ispisati -1 na standardni izlaz za greške.

NAPOMENA: *Podrazumevati da su ulazni podaci o takmičenjima ispravni. Broj osvajača nije unapred poznat.*

Test 1

```
Pokretanje: ./a.out -y 2016
Ulaz:
  2011 ManUtd
  2012 ManCity
  2013 ManUtd
  2014 ManCity
  2015 Chelsea
  2016 Leicester
  2017 Chelsea
  2018 ManCity
Izlaz:
  Leicester
```

Test 2

```
Pokretanje: ./a.out -w RealMadrid
Ulaz:
  2011 Barcelona
  2012 Chelsea
  2013 BayernMunich
  2014 RealMadrid
  2015 Barcelona
  2016 RealMadrid
  2017 RealMadrid
  2018 RealMadrid
Izlaz:
  2014 2016 2017 2018
```

Test 3

```
Pokretanje: ./a.out -s 2001
Izlaz za greške:
  -1
```

Test 4

```
Pokretanje: ./a.out -x
Izlaz za greške:
  -1
```

Test 5

```
Pokretanje: ./a.out -s 2012 2000
Izlaz za greške:
  -1
```

Test 6

```
Pokretanje: ./a.out -y 2005 -w RealMadrid
Izlaz za greške:
  -1
```

A.2 I smer

A.2.1 Praktični deo ispita, januar 2019.

Zadatak A.2.1 Napisati program koji učitava cele trocifrene brojeve sve do kraja ulaza i na standardni izlaz ispisuje one čije su cifre uređene strogo rastuće (cifre se čitaju sa leva na desno). U slučaju greške, ispisati -1 i prekinuti izvršavanje programa.

Test 1

```
Ulaz:
  -532 236 100 -555 546
Izlaz:
  236
```

Test 2

```
Ulaz:
  123 -123 321 -321
Izlaz:
  123 -123
```

Test 3

```
Ulaz:
  258 695 -1234
Izlaz:
  258 -1
```

Test 4

```
Ulaz:
  14
Izlaz:
  -1
```

Zadatak A.2.2 Napisati program koji sa standardnog ulaza učitava reč s maksimalne dužine 20 karaktera (bez belina), a zatim karakter koji predstavlja način modifikacije učitane niske:

- ukoliko je učitan karakter m, sve karaktere reči s koji su mala slova, pretvoriti u odgovarajuća velika
- ukoliko je učitan karakter v, sve karaktere reči s koji su velika slova, pretvoriti u odgovarajuća mala
- ukolko je učitan karakter o, ne menjati karaktere reči s

Na standardni izlaz ispisati nisku nakon modifikacije. U slučaju greške, ispisati -1 na standardni izlaz i prekinuti izvršavanje programa.

Test 1

```
ULAZ:
  sreca m
IZLAZ:
  SRECA
```

Test 2

```
ULAZ:
  IspiT v
IZLAZ:
  ispit
```

Test 3

```
ULAZ:
  Rec o
IZLAZ:
  Rec
```

Test 4

```
ULAZ:
  PROgram x
IZLAZ:
  -1
```

Zadatak A.2.3 Napisati program za praćenje rezultata automobilske trke. Na takmičenju učestvuje n ($n \geq 3$) takmičara u m ($m \geq 2$) trka. Program prvo učitava broj takmičara i trka, a zatim za svakog od n takmičara vreme u sekundama u svakoj od m trka. Pretpostaviti da neće biti više od 100 takmičara i 100 trka. Vremena čuvati u matrici dimenzije $n \times m$ tako da element (i, j) predstavlja vreme koje je takmičar i postigao u j-toj trci. Na standardni izlaz ispisati redne brojeve takmičara (brojeći ih od 0) koji su pobedili u trkama (bili najbrži), redom za svaku trku. Pretpostaviti da neće biti više takmičara sa istim prolaznim vremenom po trci. U slučaju greške, ispisati -1 na standardni izlaz i prekinuti izvršavanje programa.

Test 1

```
ULAZ:
  3 3
  192.9 87.8 109.102
  181.2 92.1 102.4
  151.1 87.9 118.9
IZLAZ:
  2 0 1
```

Test 2

```
ULAZ:
  3 4
  51.3 184.94 121.7 99.51
  50.9 182.71 119.2 99.2
  51.2 192.11 122.9 100.1
IZLAZ:
  1 1 1 1
```

Test 3

```
ULAZ:
  4 3
  113.5 145.2 -14.5
IZLAZ:
  -1
```

Test 4

```
ULAZ:
  4 -3
IZLAZ:
  -1
```

Zadatak A.2.4 Definisati strukturu sa nazivom *Kutija* koja sadrži dužinu, širinu i visinu kutije u centimetrima (pozitivni celi brojevi). Napisati program koji učitava pozitivan ceo broj n ($n \leq 100$), a zatim i podatke o n kutija. Nakon toga, program treba da ispiše zapreminu kutije u koju se može smestiti svaka od preostalih $n - 1$ kutija pojedinačno. Pretpostaviti da neće biti više takvih kutija, a ukoliko takva kutija ne postoji, ispisati 0. U slučaju greške, ispisati -1 na standardni izlaz i prekinuti izvršavanje programa.

NAPOMENA: *Da bi jedna kutija (sa celobrojnim dimenzijama) stala u drugu, svaka od dimenzija te kutije (dužina, širina i visina redom) mora biti manja barem 1 centimetar od odgovarajućih dimenzija druge kutije. Prilikom smeštanja jedne kutije u drugu nema obrtanja kutije.*

Test 1

```
ULAZ:
  4
  15 2 9
  185 27 12
  16 21 10
  120 12 3
IZLAZ:
  59940
```

Test 2

```
ULAZ:
  3
  9 18 2
  21 5 3
  3 15 5
IZLAZ:
  0
```

Test 3

```
ULAZ:
  -3
IZLAZ:
  -1
```

Test 4

```
ULAZ:
  3
  1 2 3
  8 9 -5
IZLAZ:
  -1
```

A.2.2 Praktični deo ispita, februar 2019.

Zadatak A.2.5 Napisati program koji učitava cele trocifrene brojeve sve do kraja ulaza i na standardni izlaz ispisuje one brojeve čija je cifra desetica jednaka aritmetičkoj sredini cifara stotina i jedinica. U slučaju greške, ispisati -1 na standardni izlaz i prekinuti izvršavanje programa.

Test 1

```
ULAZ:
  543 236 100 -555 546
IZLAZ:
  543 -555
```

Test 2

```
ULAZ:
  402 -402 103 -103
IZLAZ:
```

Test 3

```
ULAZ:
  -1234
IZLAZ:
  -1
```

Test 4

```
ULAZ:
  14
IZLAZ:
  -1
```

Zadatak A.2.6 Sa standardnog ulaza se učitava niska s maksimalne dužine 30 karaktera. Napisati program koji na standardni izlaz ispisuje dužinu najduže podniske niske s čiji su karakteri uređeni strogo rastuće po ASCII kodovima čitajući sa leva na desno.

Test 1

```
ULAZ:
  stolica
IZLAZ:
  2
```

Test 2

```
ULAZ:
  a12bcABc
IZLAZ:
  4
```

Test 3

```
ULAZ:
  PPPPPPPP
IZLAZ:
  1
```

Test 4

```
ULAZ:
  abcdefw
IZLAZ:
  7
```

Zadatak A.2.7 Sa standardnog ulaza se učitava neparan prirodan broj n ($n \leq 101$), a zatim n^2 celih brojeva koje treba sačuvati u odgovarajućoj kvadratnoj matrici. Proveriti da li je suma elemenata na glavnoj dijagonali matrice

neparna, i ako jeste, na standardni izlaz ispisati vrednost maksimalnog elementa glavne dijagonale. Ako to nije slučaj, ispisati vrednost minimalnog elementa glavne dijagonale. U slučaju greške, ispisati -1 na standardni izlaz i prekinuti izvršavanje programa.

Test 1

```
ULAZ:
 3
 15 6 7
 2 -4 -2
 3 2 6
IZLAZ:
 15
```

Test 2

```
ULAZ:
 5
 12 6 7 1 2
 2 -4 -2 2 0
 3 2 6 10 7
 3 2 6 12 5
 12 6 7 1 2
IZLAZ:
 -4
```

Test 3

```
ULAZ:
 4
IZLAZ:
 -1
```

Test 4

```
ULAZ:
 -7
IZLAZ:
 -1
```

Zadatak A.2.8 Definisati strukturu sa nazivom *Student* koja sadrži podatke o studentu: indeks studenta (pozitivan ceo broj), broj poena ostvaren na ispitu (nenegativan realan broj dvostruke tačnosti iz intervala $[0, 100]$) i oznaku učionice u kojoj je student polagao ispit (niska iz skupa ”704”, ”718”, ”rlab”i ”bim”). Napisati program koji sa standardnog ulaza učitava prirodan broj n, a zatim podatke o n studenata koji su polagali ispit iz Programiranja 1, redom, indeks, broj poena i oznaku učionice. Nakon podataka o studentima se učitava oznaka učionice za koju treba ispisati broj studenata iz te učionice koji su položili ispit. Oznaka učionice se zadaje kao niska od najviše 10 karaktera. Pretpostaviti da su podaci o studentima ispravni i da neće biti više od 100 studenata. U slučaju greške ispisati -1 na standardni izlaz i prekinuti izvršavanje programa. Student je položio ispit ako je na istom ostvario bar 51 poen.

Test 1

```
ULAZ:
 9
 20180001 98 704
 20180002 33 704
 20180003 7 718
 20180005 61.8 rlab
 20180006 50.5 bim
 20180007 55.6 718
 20180008 51 704
 20180009 30 rlab
 20180010 40.4 rlab
 704
IZLAZ:
 2
```

Test 2

```
ULAZ:
 7
 20180003 73 718
 20180005 60.8 rlab
 20180006 40.5 bim
 20180007 45.6 718
 20180008 19.9 704
 20180009 31.4 rlab
 20180010 49.4 rlab
 rlab
IZLAZ:
 1
```

Test 3

```
ULAZ:
 4
 20180001 98 704
 20180002 33 704
 20180003 73.5 718
 20180005 60.8 rlab
 bim
IZLAZ:
 0
```

Test 4

```
ULAZ:
 -4
IZLAZ:
 -1
```

A.3 Rešenja

Rešenje A.1.1

```
#include <stdio.h>
#include <stdlib.h>

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int broj, broj_kopija;
  char cifra_jedinica, cifra_desetica, cifra_stotina, cifra_hiljada;
  char postoji_broj = 0;


  while (1) {
    /* Ucitavanje korisnickog unosa. */
    scanf("%d", &broj);

    /* Provera da li se doslo do kraja unosa. */
    if (broj == 0) {
      break;
    }

    /* Cuvanje kopije broja. */
    broj_kopija = broj;

    /* Provera ispravnosti ulaza. */
    broj = abs(broj);
    if (broj < 1000 || broj > 9999) {
      fprintf(stderr, "-1\n");
      exit(EXIT_FAILURE);
    }

    /* Izdvajanje cifara zadatog broja. */
    cifra_jedinica = broj % 10;
    broj /= 10;

    cifra_desetica = broj % 10;
    broj /= 10;

    cifra_stotina = broj % 10;
    broj /= 10;

    cifra_hiljada = broj;

    /* Proverava da li je cifra desetica najveca cifra. */
    if (cifra_desetica >= cifra_jedinica
        && cifra_desetica >= cifra_stotina
        && cifra_desetica >= cifra_hiljada) {
      /* Ako jeste, ispisuje se ucitani broj. */
      printf("%d\n", broj_kopija);
```

```
48
      /* Pamti se informacija da je broj sa ovim svojstvom
50       pronadjen. */
      postoji_broj = 1;
52    }
  }
54
  /* Ako broj sa trazenim svojstvom nije pronadjen, ispisuje se
56     odgovarajuca poruka. */
  if (!postoji_broj) {
58   printf("0\n");
  }
60
  exit(EXIT_SUCCESS);
62 }
```

Rešenje A.1.2

```
#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAKS_DUZINA 101
6
int main() {
8   /* Deklaracija potrebnih promenljivih. */
  char email[MAKS_DUZINA];
10  char sifrovani_email[MAKS_DUZINA];
  char *at_pozicija;
12
  /* Ucitavanje elektronske adrese. */
14  scanf("%s", email);

16  /* Odredjivanje pozicije @ karaktera. */
  at_pozicija = strchr(email, '@');
18
  /* Ukoliko elektronska adresa ne sadrzi @ karakter, ispisuje se
20     trazena poruka. */
  if (at_pozicija == NULL) {
22    fprintf(stderr, "-1\n");
    exit(EXIT_FAILURE);
24  }

26  /* Sifrovana adresa inicijalno sadrzi samo terminirajucu nulu. */
  sifrovani_email[0] = '\0';
28
  /* U sifrovanu adresu se kopira deo originalne adrese koji
30     prethodi @ karakteru. */
  *at_pozicija = '\0';
32  strcpy(sifrovani_email, email);
```

```
34    /* Zatim se sifrovana adresa nadovezuje sa [AT] zamenom. */
      strcat(sifrovani_email, "[AT]");
36
      /* Na kraju se sifrovana adresa nadovezuje sa delom originalne
38       adrese koji se nalazi posle @ karaktera. */
      strcat(sifrovani_email, at_pozicija + 1);
40
      /* Ispis rezultata. */
42    printf("%s\n", sifrovani_email);

44    exit(EXIT_SUCCESS);
  }
```

Rešenje A.1.3

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>

5 #define MAKS_DUZINA 21
  #define MAKS_DUZINA_OZNAKE 3
7 #define MAKS_BROJ_ELEMENATA 120

9 /* Struktura koja opisuje hemijski element. */
  typedef struct Hemijski_element {
11   char naziv[MAKS_DUZINA];
     char oznaka[MAKS_DUZINA_OZNAKE];
13   int broj_neutrona;
  } Hemijski_element;
15
  int main() {
17   /* Deklaracija potrebnih promenljivih. */
     Hemijski_element elementi[MAKS_BROJ_ELEMENATA];
19   int i, n;
     char naziv_trazenog_elementa[MAKS_DUZINA];
21   char nadjen;

23   /* Ucitavanje hemijskih elemenata. */
     for (i = 0;; i++) {
25     /* Prvo se ucitava naziv elementa. */
       scanf("%s", elementi[i].naziv);
27
       /* Ako je u pitanju rec "kraj", ucitavanje hemijskih elemenata se
29        prekida. */
       if (strcmp(elementi[i].naziv, "kraj") == 0) {
31       break;
       }
33
       /* U suprotnom, ucitava se oznaka elementa i broj neutrona. */
35     scanf("%s%d", elementi[i].oznaka, &elementi[i].broj_neutrona);
     }
```

```
37
   /* Poslednja vrednost brojaca i odgovara broju elemenata ucitanog
39    niza. */
   n = i;
41
   /* Ucitavanje naziva trazenog elementa. */
43 scanf("%s", naziv_trazenog_elementa);

45 /* Provera da li se trazeni element nalazi u nizu elemenata. */
   /* Informacija da li se element nalazi u nizu ili ne bice upisana
47    kao vrednost 1 ili 0 u promenljivu nadjen. */
   nadjen = 0;
49 for (i = 0; i < n; i++) {
     if (strcmp(elementi[i].naziv, naziv_trazenog_elementa) == 0) {
51     nadjen = 1;
       printf("%s %d\n", elementi[i].oznaka,
53            elementi[i].broj_neutrona);
       break;
55   }
   }
57
   /* Ukoliko se trazeni element ne nalazi u nizu elemenata,
59    ispisuje se odgovarajuca poruka. */
   if (!nadjen) {
61   fprintf(stderr, "-1\n");
   }
63
   return 0;
65 }
```

Rešenje A.1.4

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <string.h>
   #include <ctype.h>
5
   #define MAKS_DUZINA 31
7
   /* Funkcija ispisuje odgovarajucu poruku na standardni izlaz za
9     greske i prekida izvrsavanje programa. */
   void greska() {
11   fprintf(stderr, "-1\n");
     exit(EXIT_FAILURE);
13 }

15 int main(int argc, char *argv[]) {
     /* Deklaracija potrebnih promenljivih. */
17   FILE *ulaz;
     int n, i;
19   char karakter;
```

```
  char opcija;
21 char zanr[MAKS_DUZINA], tmp_pesma[MAKS_DUZINA],
       tmp_zanr[MAKS_DUZINA];
23
  /* Proverava broja argumenata komandne linije. */
25 if (argc != 2) {
    greska();
27 }

29 /* Proverava da li je opcija ispravno zadata tj. da li pocinje
      karakterom -. */
31 if (argv[1][0] != '-') {
    greska();
33 }

35 /* Ako je opcija ispravno zadata, cuva se u promenljivoj opcija. */
   opcija = argv[1][1];
37
   /* Otvaranje datoteke za citanje i proverava uspesnosti
39    otvaranja. */
   ulaz = fopen("pesme.txt", "r");
41 if (ulaz == NULL) {
    greska();
43 }

45 /* Ucitavanje broja pesama. */
   fscanf(ulaz, "%d", &n);
47
   /* Analiza zadate opcije. */
49 switch (opcija) {

51 case 'p':
     /* 1) cita se karakter za pretragu */
53   scanf("%c", &karakter);

55   /* 2) za svaku pesmu */
     for (i = 0; i < n; i++) {
57     /* 3) citaju se ime pesme i zanr pesme */
       fscanf(ulaz, "%s", tmp_pesma);
59     fscanf(ulaz, "%s", tmp_zanr);

61     /* 4) proverava se da li ime pesme pocinje procitanim
          karakterom */
63     if (toupper(tmp_pesma[0]) == toupper(karakter)) {
         /* 5) ispisuje se ime pesme */
65       printf("%s\n", tmp_pesma);
       }
67   }
     break;
69
   case 'z':
71   /* 1) ucitava se zanr */
```

```
    scanf("%s", zanr);

    /* 2) za svaku pesmu */
    for (i = 0; i < n; i++) {
      /* 3) citaju se ime pesme i zanr pesme */
      fscanf(ulaz, "%s", tmp_pesma);
      fscanf(ulaz, "%s", tmp_zanr);

      /* 4) proverava se da li zanr pesme odgovara procitanom zanru
       */
      if (strcmp(tmp_zanr, zanr) == 0) {
        /* 5) ispisuje se ime pesme */
        printf("%s\n", tmp_pesma);
      }
    }
    break;

  default:
    /* Ako je zadata pogresna opcija, prekida se izvrsavanje
       programa. */
    greska();
  }

  /* Zatvaranje datoteke. */
  fclose(ulaz);

  exit(EXIT_SUCCESS);
}
```

Rešenje A.1.5

```
#include <stdio.h>
#include <stdlib.h>

/* Funkcija ispisuje odgovarajucu poruku na standardni izlaz za
   greske i prekida izvrsavanje programa. */
void greska() {
  fprintf(stderr, "-1\n");
  exit(EXIT_FAILURE);
}

int main() {
  /* Deklaracije potrebnih promenljivih. */
  int n;
  char cifra_jedinica, cifra_desetica, cifra_stotina, cifra_hiljada;
  float aritmeticka_sredina;
  char suma_cifara;

  /* Ucitavanje i provera ispravnosti ulaza. */
  scanf("%d", &n);
  if (n < 1000 || n > 9999) {
```

```
    greska();
  }

  /* Izdvajanje cifara unetog broja. */
  cifra_jedinica = n % 10;
  cifra_desetica = (n / 10) % 10;
  cifra_stotina = (n / 100) % 10;
  cifra_hiljada = n / 1000;

  /* Izracunavanje aritmeticke sredine cifara. */
  aritmeticka_sredina =
      (cifra_hiljada + cifra_desetica + cifra_jedinica +
       cifra_stotina) / 4.0;

  /* Izracunavanje sume onih cifara koje su vece od aritmeticke
     sredine. */
  suma_cifara = 0;

  if (cifra_jedinica > aritmeticka_sredina)
    suma_cifara += cifra_jedinica;

  if (cifra_desetica > aritmeticka_sredina)
    suma_cifara += cifra_desetica;

  if (cifra_stotina > aritmeticka_sredina)
    suma_cifara += cifra_stotina;

  if (cifra_hiljada > aritmeticka_sredina)
    suma_cifara += cifra_hiljada;

  /* Ispis rezultata. */
  printf("%d\n", suma_cifara);

  exit(EXIT_SUCCESS);
}
```

Rešenje A.1.6

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 21

/* Funkcija ispisuje odgovarajucu poruku na standardni izlaz za
   greske i prekida izvrsavanje programa. */
void greska() {
  fprintf(stderr, "-1\n");
  exit(EXIT_FAILURE);
}
```

```
14 int main() {
     /* Deklaracije potrebnih promenljivih. */
16   char s[MAX];
     char novo_s[MAX];
18   int n;

20   /* Ucitavanje niske. */
     scanf("%s", s);
22
     /* Provera duzine ucitane niske. */
24   n = strlen(s);
     if (n % 2 != 0) {
26     greska();
     }
28
     /* Popunjavanje nove niske nulama. */
30   memset(novo_s, '\0', sizeof(n));

32   /* Kopiranje druge polovine niske s u novu nisku. */
     strcpy(novo_s, s + n / 2);
34
     /* Kopiranje prve polovine niske s u novu nisku. */
36   s[n / 2] = 0;
     strcpy(novo_s + n / 2, s);
38
     /* Ispis rezultata. */
40   printf("%s\n", novo_s);

42   exit(EXIT_SUCCESS);
   }
```

Rešenje A.1.7

```
1  #include <stdio.h>
   #include <stdlib.h>
3  #include <ctype.h>
   #include <string.h>
5
   #define MAX 21
7
   /* Funkcija ispisuje odgovarajucu poruku na standardni izlaz za
9     greske i prekida izvrsavanje programa. */
   void greska() {
11   fprintf(stderr, "-1\n");
     exit(EXIT_FAILURE);
13 }

15 /* Funkcija proverava da li je niska s zapisana samo pomocu
      cifara. Povratna vrednost funkcije je 1 ako je uslov ispunjen,
17    dok je u suprotnom 0. */
   int sve_cifre(const char *s) {
```

```
19    int i;

21    /* Provera da je pocetni karakter niske znak -. */
      if (s[0] == '-') {
23      if (strlen(s) == 1)
          return 0;
25      else
          s += 1;
27    }

29    /* Provera da li su karakteri niske cifre: cim se pronadje
         karakter koji nije cifra, izvrsavanje funkcije se prekida. */
31    i = 0;
      while (s[i]) {
33      if (!isdigit(s[i]))
          return 0;
35
        i++;
37    }

39    return 1;
  }
41
  int main() {
43    /* Deklaracije potrebnih promenljivih. */
      FILE *ulaz = NULL;
45    char s[MAX];

47    /* Otvaranje datoteke za citanje i proverava uspesnosti
         otvaranja. */
49    if ((ulaz = fopen("ulaz.txt", "r")) == NULL)
        greska();
51
      /* Citaju se niske datoteke sve do kraja ulaza. */
53    while (fscanf(ulaz, "%s", s) != EOF)
        /* Ako se procitana niska sastoji samo od brojeva, ispisuje se
55         na standardni izlaz. */
        if (sve_cifre(s))
57        printf("%s ", s);

59    putchar('\n');

61    /* Zatvaranje datoteke. */
      fclose(ulaz);
63
      exit(EXIT_SUCCESS);
65  }
```

Rešenje A.1.8

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 31

/* Funkcija ispisuje odgovarajucu poruku na standardni izlaz za
   greske i prekida izvrsavanje programa. */
void greska() {
  fprintf(stderr, "-1\n");
  exit(EXIT_FAILURE);
}

int main(int argc, char *argv[]) {
  /* Deklaracije potrebnih promenljivih. */
  int godina;
  int tekuca_godina;
  char ime[MAX];
  char tekuce_ime[MAX];

  /* Proverava broja argumenata komandne linije. */
  if (argc != 3) {
    greska();
  }

  /* Provera da li je prvi argument komandne linije -y. */
  if (!strcmp(argv[1], "-y")) {
    /* Ako jeste, ocitava se godina koja se ocekuje kao drugi
       argument. */
    godina = atoi(argv[2]);

    /* Sve do kraja unosa ucitavaju se podaci o osvajacima. */
    while (scanf("%d %s", &tekuca_godina, tekuce_ime) == 2) {
      /* Ako uneta godina odgovara trazenoj godini, ispisuje se ime
         osvajaca. */
      if (tekuca_godina == godina) {
        printf("%s\n", tekuce_ime);
      }
    }

    exit(EXIT_SUCCESS);
  }

  /* Provera da li je prvi argument komandne linije -w. */
  if (!strcmp(argv[1], "-w")) {
    /* Ako jeste, ocitava se ime osvajaca koje se ocekuje kao drugi
       argument. */
    strcpy(ime, argv[2]);

    /* Sve do kraja unosa ucitavaju se podaci o osvajacima. */
```

```
    while (scanf("%d %s", &tekuca_godina, tekuce_ime) == 2) {
52    /* Ako uneto ime odgovara imenu osvajaca, ispisuje se godina.
       */
54    if (!strcmp(ime, tekuce_ime)) {
        printf("%d ", tekuca_godina);
56    }
    }
58  putchar('\n');

60  exit(EXIT_SUCCESS);
  }
62
  /* Ako prvi argument komandne linije nije ni -y ni -w, program
64   nije korektno pozvan. */
  greska();
66
}
```

Rešenje A.2.1

```
#include <stdio.h>
2 #include <stdlib.h>

4 int main() {
  /* Deklaracija potrebnih promenljivih. */
6 int x, abs_x;
  int cifra_jedinica, cifra_desetica, cifra_stotina;
8
  /* Ucitavanje brojeva sve do kraja ulaza. */
10 while (scanf("%d", &x) != EOF) {

12  /* Izracunavanje apsolutne vrednosti tekuceg broja. */
    abs_x = x < 0 ? -x : x;
14
    /* Provera da li je u pitanju trocifren broj. */
16  if (abs_x < 100 || abs_x > 999) {
      printf("-1\n");
18    exit(EXIT_FAILURE);
    }
20
    /* Izdvajanje cifara broja. */
22  cifra_jedinica = abs_x % 10;
    cifra_desetica = (abs_x / 10) % 10;
24  cifra_stotina = abs_x / 100;

26  /* Provera da li su cifre broja uredjene rastuce. */
    if (cifra_jedinica > cifra_desetica
28      && cifra_desetica > cifra_stotina) {
      printf("%d ", x);
30    }
  }
```

```
32    printf("\n");

34    exit(EXIT_SUCCESS);
  }
```

Rešenje A.2.2

```
  #include <stdio.h>
2 #include <string.h>
  #include <ctype.h>
4 #include <stdlib.h>

6 #define MAX 21

8 int main() {
    /* Deklaracija potrebnih promenljivih. */
10  char s[MAX], c;
    int i, n;
12
    /* Ucitavanje reci i karaktera koji odredjuje tip transformacije.
14   */
    scanf("%s %c", s, &c);
16
    /* Odredjivanje duzine reci. */
18  n = strlen(s);

20  /* Analiza procitanog karaktera. */
    switch (c) {
22  case 'm':
      /* Zamena svih malih slova reci odgovarajucim velikim slovima. */
24    for (i = 0; i < n; i++) {
        if (islower(s[i])) {
26        s[i] = toupper(s[i]);
        }
28    }
      break;
30
    case 'v':
32    /* Zamena svih velikih slova reci odgovarajucim malim slovima. */
      for (i = 0; i < n; i++) {
34      if (isupper(s[i])) {
          s[i] = tolower(s[i]);
36      }
      }
38    break;

40  case 'o':
      /* Rec se ne menja. */
42    break;
    default:
44    /* Transformacija nije definisana pa se ispisuje poruka o
```

```
        gresci i prekida izvrsavanje programa. */
46    printf("-1\n");
      exit(EXIT_FAILURE);
48  }

50  /* Ispis rezultata. */
    printf("%s\n", s);
52
    exit(EXIT_SUCCESS);
54 }
```

Rešenje A.2.3

```
#include <stdio.h>
2 #include <stdlib.h>

4 #define MAX 100

6 int main() {
    /* Deklaracija potrebnih promenljivih. */
8   int n, m, i, j, redni_broj_pobednika;
    float trke[MAX][MAX], pobednik;
10
    /* Ucitavanje broja takmicara i broja trka. */
12  scanf("%d%d", &n, &m);

14  /* Provera ispravnosti ucitanih vrednosti. */
    if (n < 3 || n > MAX || m < 2 || m > MAX) {
16    printf("-1\n");
      exit(EXIT_FAILURE);
18  }

20  /* Ucitavanje vremena takmicara po trkama. */
    for (i = 0; i < n; i++) {
22    for (j = 0; j < m; j++) {
        scanf("%f", &trke[i][j]);
24
        /* Provera da li je zadato korektno vreme. */
26      if (trke[i][j] <= 0.0) {
          printf("-1\n");
28        exit(EXIT_FAILURE);
        }
30    }
    }
32
    /* Odredjivanje pobednika u trkama. */
34  for (j = 0; j < m; j++) {

36    /* Odredjivanje pobednika j-te trke se svodi na problem
         pronalazenje minimuma j-te kolone. */
38    pobednik = trke[0][j];
```

```
    redni_broj_pobednika = 0;

    for (i = 1; i < n; i++) {
      if (pobednik > trke[i][j]) {
        pobednik = trke[i][j];
        redni_broj_pobednika = i;
      }
    }

    printf("%d ", redni_broj_pobednika);
  }

  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje A.2.4

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

/* Struktura koja opisuje kutiju. */
typedef struct {
  int sirina, duzina, visina;
} Kutija;

/* Funkcija proverava da li se u zadatu kutiju koja se nalazi
   u nizu kutija na poziciji j mogu smestiti preostale kutije.
   Povratna vrednost funkcije je 1 ako kutije mogu da se smeste, a
   u suprotnom 0. */
int smesti(Kutija kutije[], int n, Kutija * kutija, int j) {
  int i;

  /* Uporedjuju se dimenzije zadate j-te kutije sa dimenzijama svih
     preostalih kutija. */
  for (i = 0; i < n; i++) {
    if (i != j
        && (kutije[i].sirina >= kutija->sirina
            || kutije[i].duzina >= kutija->duzina
            || kutije[i].visina >= kutija->visina)) {
      return 0;
    }
  }

  return 1;
}

int main() {
```

```
  /* Deklaracija potrebnih promenljivih. */
34  Kutija kutije[MAX];
    int i, n;
36
    /* Ucitavanje broja kutija i provera ispravnosti ulaza. */
38  scanf("%d", &n);
    if (n <= 0 || n > MAX) {
40    printf("-1\n");
      exit(EXIT_FAILURE);
42  }

44  /* Ucitavanje dimenzija kutija uz proveru ispravnosti ulaza. */
    for (i = 0; i < n; i++) {
46    scanf("%d%d%d", &kutije[i].sirina, &kutije[i].duzina,
            &kutije[i].visina);
48    if (kutije[i].sirina <= 0 || kutije[i].duzina <= 0
          || kutije[i].visina <= 0) {
50      printf("-1\n");
        exit(EXIT_FAILURE);
52    }
    }
54
    /* Za svaku kutiju se proverava trazeno svojstvo. */
56  for (i = 0; i < n; i++) {
      /* Ukoliko u i-tu kutiju mogu da se smeste preostale kutije,
58       izracunava se i ispisuje njena zapremina. */
      if (smesti(kutije, n, &kutije[i], i)) {
60      printf("%d\n",
                kutije[i].sirina * kutije[i].duzina *
62              kutije[i].visina);
        exit(EXIT_SUCCESS);
64    }
    }
66
    /* U suprotnom, zakljucujemo da ne postoji kutija sa trazenim
68     svojstvom. */
    printf("0\n");
70
    exit(EXIT_SUCCESS);
72  }
```

Rešenje A.2.5

```
  #include <stdio.h>
2 #include <stdlib.h>

4 int main() {
    /* Deklaracija potrebnih promenljivih. */
6   int x, abs_x;
    int cifra_jedinica, cifra_desetica, cifra_stotina;
8
```

```
  /* Ucitavanje brojeva sve do kraja ulaza. */
  while (scanf("%d", &x) != EOF) {

    /* Izracunavanje apsolutne vrednosti tekuceg broja. */
    abs_x = x < 0 ? -x : x;

    /* Provera da li je u pitanju trocifren broj. */
    if (abs_x < 100 || abs_x > 999) {
      printf("-1\n");
      exit(EXIT_FAILURE);
    }

    /* Izdvajanje cifara broja. */
    cifra_jedinica = abs_x % 10;
    cifra_desetica = (abs_x / 10) % 10;
    cifra_stotina = abs_x / 100;

    /* Provera da li je cifra desetica jednaka aritmetickoj sredini
       cifara stotine i jedinice. */
    if (cifra_desetica == (cifra_jedinica + cifra_stotina) / 2.0) {
      printf("%d ", x);
    }
  }
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje A.2.6

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 31

int main() {
  /* Deklaracija potrebnih promenljivih. */
  char s[MAX];
  int i, max_duzina, trenutna_duzina;

  /* Ucitavanje niske. */
  scanf("%s", s);

  /* Odredjivanje najduze podniske karaktera koji su uredjeni
     rastuce. */
  max_duzina = 1;
  trenutna_duzina = 1;

  for (i = 1; s[i]; i++) {
    /* Ako je ASCII kod tekuceg karaktera veci od ASCII koda
       prethodnog karaktera, podniska je rastuca pa se njena
```

```
22        trenutna duzina uvecava. */
      if (s[i - 1] < s[i]) {
24        trenutna_duzina++;
      } else {
26        /* Ako se naislo na par karaktera koji nisu uredjeni rastuce,
             azurira se, po potrebi, maksimalna duzina trazene podniske
28           i resetuje se trenutna duzina. */
          if (max_duzina < trenutna_duzina) {
30          max_duzina = trenutna_duzina;
          }
32        trenutna_duzina = 1;
      }
34    }

36    /* Postupak azuriranja maksimalne duzine se, po potrebi, vrsi i
         kada se stigne do kraja niske. */
38    if (max_duzina < trenutna_duzina) {
        max_duzina = trenutna_duzina;
40    }

42    /* Ispis rezultata. */
      printf("%d\n", max_duzina);
44
      exit(EXIT_SUCCESS);
46  }
```

Rešenje A.2.7

```
  #include <stdio.h>
2 #include <stdlib.h>

4 #define MAX 101

6 /* Funkcija izracunava zbir elemenata na glavnoj dijagonali
     kvadratne matrice dimenzije n. */
8 int suma(int m[][MAX], int n) {
    int i, s = 0;
10
    for (i = 0; i < n; i++) {
12    s += m[i][i];
    }
14
    return s;
16 }

18 /* Funkcija izracunava vrednost najmanjeg elementa glavne
     dijagonale kvadratne matrice dimenzije n. */
20 int minimum(int m[][MAX], int n) {
    int i;
22  int min = m[0][0];
```

```
  for (i = 1; i < n; i++) {
    if (min > m[i][i]) {
      min = m[i][i];
    }
  }

  return min;
}

/* Funkcija izracunava vrednost najveceg elementa glavne
   dijagonale kavdratne matrice dimenzije n. */
int maximum(int m[][MAX], int n) {
  int i;
  int max = m[0][0];

  for (i = 1; i < n; i++) {
    if (max < m[i][i]) {
      max = m[i][i];
    }
  }

  return max;
}

int main() {
  /* Deklaracija potrebnih promenljivih. */
  int m[MAX][MAX], n, i, j;

  /* Ucitavanje dimenzije matrice i provera ispravnosti ulaza. */
  scanf("%d", &n);
  if (n <= 0 || n > MAX || n % 2 == 0) {
    printf("-1\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje elemenata matrice. */
  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++) {
      scanf("%d", &m[i][j]);
    }
  }

  /* Provera da li je suma elemenata na glavnoj dijagonali matrice
     parna. */
  if (suma(m, n) % 2) {
    /* Ako jeste, ispisuje se vrednost maksimalnog elementa
       dijagonale. */
    printf("%d\n", maximum(m, n));
  } else {
    /* U suprotnom se ispisuje vrednost minimalnog elementa
       dijagonale. */
    printf("%d\n", minimum(m, n));
```

```
  }

  exit(EXIT_SUCCESS);
}
```

Rešenje A.2.8

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAX_BROJ_STUDENATA 100
#define MAX_UCIONICA 11

/* Struktura koja opisuje studenta. */
typedef struct {
  int indeks;
  double poeni;
  char ucionica[MAX_UCIONICA];
} STUDENT;

int main() {
  /* Deklaracija potrebnih promenljivih. */
  STUDENT studenti[MAX_BROJ_STUDENATA];
  int i, n, broj_studenata;
  char ucionica[MAX_UCIONICA];

  /* Ucitavanje broja studenata i provera ispravnosti ulaza. */
  scanf("%d", &n);
  if (n <= 0 || n > MAX_BROJ_STUDENATA) {
    printf("-1\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitavanje podataka o studentima. */
  for (i = 0; i < n; i++) {
    scanf("%d%lf%s", &studenti[i].indeks, &studenti[i].poeni,
          studenti[i].ucionica);
  }

  /* Ucitavanje oznake ucionice i provera ispravnosti ulaza. */
  scanf("%s", ucionica);
  if (strcmp(ucionica, "704") && strcmp(ucionica, "718") &&
      strcmp(ucionica, "bim") && strcmp(ucionica, "rlab")) {
    printf("-1\n");
    exit(EXIT_FAILURE);
  }

  /* Odredjivanje broja studenata koji su polagali ispit u zadatoj
     ucionici i polozili ga. */
  broj_studenata = 0;
```

```
  for (i = 0; i < n; i++) {
46  if (!strcmp(ucionica, studenti[i].ucionica)
        && studenti[i].poeni >= 51.0) {
48    broj_studenata++;
    }
50  }

52  /* Ispis rezultata. */
  printf("%d\n", broj_studenata);
54
  exit(EXIT_SUCCESS);
56}
```