### PROGRAMIRANJE 2

## Milena Vujošević Janičić, Jelena Graovac, Nina Radojičić, Ana Spasić, Mirko Spasić, Anđelka Zečević

## PROGRAMIRANJE 2 Zbirka zadataka sa rešenjima

Beograd 2016.

#### Autori:

dr Milena Vujošević Janičić, docent na Matematičkom fakultetu u Beogradu dr Jelena Graovac, docent na Matematičkom fakultetu u Beogradu Nina Radojičić, asistent na Matematičkom fakultetu u Beogradu Ana Spasić, asistent na Matematičkom fakultetu u Beogradu Mirko Spasić, asistent na Matematičkom fakultetu u Beogradu Anđelka Zečević, asistent na Matematičkom fakultetu u Beogradu

#### PROGRAMIRANJE 2

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 i crteži: autori. Dizajn korica: Anđelka Zečević

Štampa: Copy Centar, Beograd. Tiraž 200.

СІР Каталогизација у публикацији

Народна библиотека Србије, Београд

004.4(075.8)(076)

004.432.2C(075.8)(076)

PROGRAMIRANJE 2 : zbirka zadataka sa rešenjima / Milena Vujošević

Janičić ... [et al.]. - Beograd : Matematički fakultet, 2016

(Beograd: Copy Centar). - VII, 361 str.; 24 cm

Tiraž 200.

ISBN 978-86-7589-107-9

- 1. Вујошевић Јаничић, Милена 1980- [аутор]
- а) Програмирање Задаци b) Програмски језик "С"- Задаци COBISS.SR-ID 221508876

©2016. Milena Vujošević Janičić, Jelena Graovac, Nina Radojičić, Ana Spasić, Mirko Spasić, Anđelka Zečević

Ovo delo zaštićeno je licencom Creative Commons CC BY-NC-ND 4.0 (Attribution-NonCommercial-NoDerivatives 4.0 International License). Detalji licence mogu se videti na veb-adresi http://creativecommons.org/licenses/by-nc-nd/4.0/. Dozvoljeno je umnožavanje, distribucija i javno saopštavanje dela, pod uslovom da se navedu imena autora. Upotreba dela u komercijalne svrhe nije dozvoljena. Prerada, preoblikovanje i upotreba dela u sklopu nekog drugog nije dozvoljena.



# Sadržaj

1	Uvo	odni zadaci 1	L
	1.1	Podela koda po datotekama	Ĺ
	1.2	Algoritmi za rad sa bitovima	
	1.3	Rekurzija	
	1.4	Rešenja	3
2	Pok	azivači 67	7
	2.1	Pokazivačka aritmetika	7
	2.2	Višedimenzioni nizovi	Ĺ
	2.3	Dinamička alokacija memorije	ó
	2.4	Pokazivači na funkcije	
	2.5	Rešenja	3
3	Alg	oritmi pretrage i sortiranja 125	ó
	3.1	Algoritmi pretrage	<i>.</i>
	3.2	Algoritmi sortiranja	
	3.3	Bibliotečke funkcije pretrage i sortiranja	
	3.4	Rešenja	
4	Din	amičke strukture podataka 217	7
_	4.1	Liste	7
	4.2	Stabla	
	4.3	Rešenja	
A	Ispi	tni rokovi 333	3
		Praktični deo ispita, jun 2015	3
		Praktični deo ispita, jul 2015	
		Praktični deo ispita, septembar 2015	
		Praktični deo ispita, januar 2016	
		Rosonia 340	

## Predgovor

U okviru kursa *Programiranje 2* na Matematičkom fakultetu vežbaju se zadaci koji imaju za cilj da studente nauče rekurzivnom pristupu rešavanja problema, ispravnom radu sa pokazivačima i dinamički alociranom memorijom, osnovnim algoritmima pretraživanja i sortiranja, kao i radu sa dinamičkim strukturama podataka, poput listi i stabala. Zadaci koji se nalaze u ovoj zbirci predstavljaju objedinjen skup zadataka sa vežbi i praktikuma ovog kursa, kao i primere zadataka sa održanih ispita. Elektronska verzija zbirke i propratna rešenja u elektronskom formatu, dostupna su besplatno u okviru strane kursa www.programiranje2.matf.bg.ac.rs u skladu sa navedenom licencom.

U prvom poglavlju zbirke obrađene su uvodne teme koje obuhvataju osnovne tehnike koje se koriste u rešavanju svih ostalih zadataka u zbirci: podela koda po datotekama i rekurzivni pristup rešavanju problema. Takođe, u okviru ovog poglavlja dati su i osnovni algoritmi za rad sa bitovima. Drugo poglavlje je posvećeno pokazivačima: pokazivačkoj aritmetici, višedimenzionim nizovima, dinamičkoj alokaciji memorije i radu sa pokazivačima na funkcije. Treće poglavlje obrađuje algoritme pretrage i sortiranja, a četvrto dinamičke strukture podataka: liste i stabla. Dodatak sadrži najvažnije ispitne rokove 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 u prethodnih desetak 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 korišćni za početno testiranje (koje prethodi ocenjivanju) studentskih radova na ispitima.

Neizmerno zahvaljujemo recenzentima, Gordani Pavlović Lažetić i Draganu Uroševiću, na veoma pažljivom čitanju rukopisa i na brojnim korisnim sugestijama. Takođe, zahvaljujemo studentima koji su svojim aktivnim učešćem u nastavi pomogli i doprineli uobličavanju ovog materijala.

Svi komentari i sugestije na sadržaj zbirke su dobrodošli i osećajte se slobodnim da ih pošaljete elektronskom poštom bilo kome od autora<sup>1</sup>.

Autori

 $<sup>^1\</sup>mathrm{Adrese}$ autora su: milena, j<br/>graovac, nina, aspasic, mirko, andjelkaz, sa nastavkom <br/>  $\mathtt{Cmatf.bg.ac.rs}$ 

## 1

## Uvodni zadaci

### 1.1 Podela koda po datotekama

Zadatak 1.1 Napisati program za rad sa kompleksnim brojevima.

- (a) Definisati strukturu KompleksanBroj koja opisuje kompleksan broj zadat njegovim realnim i imaginarnim delom.
- (b) Napisati funkciju void ucitaj\_kompleksan\_broj(KompleksanBroj \* z) koja učitava kompleksan broj z sa standardnog ulaza.
- (c) Napisati funkciju void ispisi\_kompleksan\_broj(KompleksanBroj z) koja ispisuje kompleksan broj z na standardni izlaz u odgovarajućem formatu.
- (d) Napisati funkciju float realan\_deo(KompleksanBroj z) koja vraća vrednost realnog dela broja z.
- (e) Napisati funkciju float imaginaran\_deo(KompleksanBroj z) koja vraća vrednost imaginarnog dela broja z.
- (f) Napisati funkciju float moduo (KompleksanBroj z) koja vraća moduo kompleksnog broja z.
- (g) Napisati funkciju KompleksanBroj konjugovan(KompleksanBroj z) koja vraća konjugovano-kompleksni broj broja z.
- (h) Napisati funkciju KompleksanBroj saberi (KompleksanBroj z1, KompleksanBroj z2) koja vraća zbir dva kompleksna broja z1 i z2.

- (i) Napisati funkciju KompleksanBroj oduzmi (KompleksanBroj z1, KompleksanBroj z2) koja vraća razliku dva kompleksna broja z1 i z2.
- (j) Napisati funkciju KompleksanBroj mnozi (KompleksanBroj z1, KompleksanBroj z2) koja vraća proizvod dva kompleksna broja z1 i z2.
- (k) Napisati funkciju float argument(KompleksanBroj z) koja vraća argument kompleksnog broja z.

Napisati program koji testira prethodno napisane funkcije. Sa standardnog ulaza uneti dva kompleksna broja z1 i z2, a zatim ispisati realni deo, imaginarni deo, moduo, konjugovano-kompleksan broj i argument broja koji se dobija kao zbir, razlika ili proizvod brojeva z1 i z2 u zavisnosti od znaka ('+', '-', '\*') koji se unosi sa standardnog ulaza.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:

Unesite realni i imaginarni deo kompleksnog broja: 1 -3
(1.00 - 3.00 i)

Unesite realni i imaginarni deo kompleksnog broja: -1 4
(-1.00 + 4.00 i)

Unesite znak (+,-,*): -
(1.00 - 3.00 i) - (-1.00 + 4.00 i) = (2.00 - 7.00 i)

Realni_deo: 2

Imaginarni_deo: -7.000000

Moduo: 7.280110

Konjugovano kompleksan broj: (2.00 + 7.00 i)

Argument kompleksnog broja: -1.292497
```

Zadatak 1.2 Uraditi prethodni zadatak tako da su sve napisane funkcije za rad sa kompleksnim brojevima zajedno sa definicijom strukture KompleksanBroj izdvojene u posebnu biblioteku. Napisati program koji testira ovu biblioteku. Sa standardnog ulaza uneti kompeksan broj, a zatim na standardni izlaz ispisati njegov polarni oblik.

#### Primer 1

```
| INTERAKCIJA SA PROGRAMOM:
| Unesite realni i imaginarni deo kompleksnog broja: -5 2
| Polarni oblik kompleksnog broja je 5.39 * e^i * 2.76
```

#### Zadatak 1.3 Napisati biblioteku za rad sa polinomima.

(a) Definisati strukturu Polinom koja opisuje polinom stepena najviše 20 koji je zadat nizom svojih koeficijenata tako da se na i-toj poziciji u nizu nalazi koeficijent uz i-ti stepen polinoma.

- (b) Napisati funkciju void ispisi(const Polinom \* p) koja ispisuje polinom p na standardni izlaz, od najvišeg ka najnižem stepenu. Ipisati samo koeficijente koji su različiti od nule.
- (c) Napisati funkciju Polinom ucitaj() koja učitava polinom sa standardnog ulaza. Za polinom najpre uneti stepen, a zatim njegove koeficijente.
- (d) Napisati funkciju double izracunaj (const Polinom \* p, double x) koja vraća vrednosti polinoma p u datoj tački x koristeći Hornerov algoritam.
- (e) Napisati funkciju Polinom saberi (const Polinom \* p, const Polinom \* q) koja vraća zbir dva polinoma p i q.
- (f) Napisati funkciju Polinom pomnozi(const Polinom \* p, const Polinom \* q) koja vraća proizvod dva polinoma p i q.
- (g) Napisati funkciju Polinom izvod(const Polinom \* p) koja vraća izvod polinoma p.
- (h) Napisati funkciju Polinom n\_izvod(const Polinom \* p, int n) koja vraća n-ti izvod polinoma p.

Napisati program koji testira prethodno napisane funkcije. Sa standardnog ulaza učitati polinome p i q, a zatim ih ispisati na standardni izlaz u odgovarajućem formatu. Izračunati i ispisati zbir  $\mathbf z$  i proizvod  $\mathbf r$  unetih polinoma  $\mathbf p$  i  $\mathbf q$ . Sa standardnog ulaza učitati realni broj  $\mathbf x$ , a zatim na standardni izlaz ispisati vrednost polinoma  $\mathbf z$  u tački  $\mathbf x$  zaokruženu na dve decimale. Na kraju, sa standardnog ulaza učitati broj  $\mathbf n$  i na izlaz ispisati  $\mathbf n$ -ti izvod polinoma  $\mathbf r$ .

```
INTERAKCIJA SA PROGRAMOM:
Unesite polinom p (prvo stepen, pa zatim koeficijente od najveceg stepena do nultog):
3 1.2 3.5 2.1 4.2
Unesite polinom q (prvo stepen, pa zatim koeficijente od najveceg stepena do nultog):
2 2.1 0 -3.9
Zbir polinoma je polinom z:
1.20x^3+5.60x^2+2.10x+0.30
Prozvod polinoma je polinom r:
2.52x^5+7.35x^4-0.27x^3-4.83x^2-8.19x-16.38
Unesite tacku u kojoj racunate vrednost polinoma z:
0
Vrednost polinoma z u tacki 0.00 je 0.30
Unesite izvod polinoma koji zelite:
3
3. izvod polinoma r je: 151.20x^2+176.40x-1.62
```

#### Zadatak 1.4 Napisati biblioteku za rad sa razlomcima.

- (a) Definisati strukturu Razlomak koja opisuje razlomak.
- (b) Napisati funkciju Razlomak ucitaj() za učitavanje razlomka.
- (c) Napisati funkciju void ispisi(const Razlomak \* r) koja ispisuje razlomak r.
- (d) Napisati funkciju int brojilac(const Razlomak \* r) koja vraćaja brojilac razlomka r.
- (e) Napisati funkciju int imenilac(const Razlomak \* r) koja vraćaja imenilac razlomka r.
- (f) Napisati funkciju double realna\_vrednost(const Razlomak \* r) koja vraća odgovarajuću realnu vrednost razlomka r.
- (g) Napisati funkciju double reciprocna\_vrednost(const Razlomak \* r) koja vraća recipročnu vrednost razlomka r.
- (h) Napisati funkciju Razlomak skrati(const Razlomak \* r) koja vraća skraćenu vrednost datog razlomka r.
- (i) Napisati funkciju Razlomak saberi(const Razlomak \* r1, const Razlomak \* r2) koja vraća zbir dva razlomka r1 i r2.
- (j) Napisati funkciju Razlomak oduzmi (const Razlomak \* r1, const Razlomak \* r2) koja vraća razliku dva razlomka r1 i r2.
- (k) Napisati funkciju Razlomak pomnozi (const Razlomak \* r1, const Razlomak \* r2) koja vraća proizvod dva razlomka r1 i r2.
- (l) Napisati funkciju Razlomak podeli(const Razlomak \* r1, const Razlomak \* r2) koja vraća količnik dva razlomka r1 i r2.

Napisati program koji testira prethodne funkcije. Sa standardnog ulaza učitati dva razlomka r1 i r2. Na standardni izlaz ispisati skraćene vrednosti zbira, razlike, proizvoda i količnika razlomaka r1 i recipročne vrednosti razlomka r2.

```
INTERAKCIJA SA PROGRAMOM:
Unesite imenilac i brojilac prvog razlomka: 1 2
Unesite imenilac i brojilac drugog razlomka: 2 3

1/2 + 3/2 = 2

1/2 - 3/2 = -1

1/2 * 3/2 = 3/4

1/2 / 3/2 = 1/3
```

## 1.2 Algoritmi za rad sa bitovima

Zadatak 1.5 Napisati biblioteku stampanje\_bitova za rad sa bitovima. Biblioteka treba da sadrži funkcije stampanje\_bitova, stampanje\_bitova\_short i stampanje\_bitova\_char za štampanje bitova u binarnom zapisu celog broja tipa int, short i char, koji se zadaje kao argument funkcije. Napisati program koji testira napisanu biblioteku. Sa standardnog ulaza učitati u heksadekadnom formatu cele brojeve tipa int, short i char i na standardni izlaz ispisati njihovu binarnu reprezentaciju.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:

Unesite broj tipa int: Ox4f4f4f4f

Binarna reprezentacija: O1001111010011110100111101001111

Unesite broj tipa short: Ox4f4f

Binarna reprezentacija: O100111101001111

Unesite broj tipa char: Ox4f

Binarna reprezentacija: 01001111
```

**Zadatak 1.6** Napisati funkcije \_bitove\_1 i prebroj\_bitove\_2 koje vraćaju broj jedinica u binarnom zapisu označenog celog broja x koji se zadaje kao argument funkcije. Prebrojavanje bitova ostvariti na dva načina:

- (a) formiranjem odgovarajuće maske i njenim pomeranjem (funkcija prebroj-\_bitove\_1)
- (b) formiranjem odgovarajuće maske i pomeranjem promenljive x (funkcija prebroj\_bitove\_2).

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati ceo broj u heksadekasnom formatu i redni broj funkcije koju treba primeniti (1 ili 2), a zatim na standardni izlaz ispisati broj jedinica u binarnom zapisu učitanog broja pozivom izabrane funkcije. Ukoliko korisnik ne unese ispravnu vrednost za redni broj funkcije, prekinuti izvršavanje programa i ispisati odgovarajuću poruku na standardni izlaz za greške.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj: Ox7F
Unesite redni broj funkcije: 1
Poziva se funkcija prebroj_bitove_1
Broj jedinica u zapisu je 7
```

#### Primer 3

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj: OxOOFFOOFF
Unesite redni broj funkcije: 2
Poziva se funkcija prebroj_bitove_2
Broj jedinica u zapisu je 16
```

#### Primer 2

```
| INTERAKCIJA SA PROGRAMOM:
Unesite broj: -0x7F
Unesite redni broj funkcije: 2
Poziva se funkcija prebroj_bitove_2
Broj jedinica u zapisu je 26
```

```
| INTERAKCIJA SA PROGRAMOM:
Unesite broj: OxOOFFOOFF
Unesite redni broj funkcije: 3
| IZLAZ ZA GREŠKE:
Greska: Neodgovarajuci redni broj funkcije.
```

Zadatak 1.7 Napisati funkcije unsigned najveci (unsigned x) i unsigned najmanji (unsigned x) koje vraćaju najveći, odnosno najmanji neoznačen ceo broj koji se može zapisati istim binarnim ciframa kao broj x.

Napisati program koji testira prethodno napisane funkcije. Sa standardnog ulaza učitati neoznačen ceo broj u heksadekadnom formatu, a zatim ispisati binarnu reprezentaciju najvećeg i najmanjeg broja koji se može zapisati istim binarnim ciframa kao učitani broj.

```
Test 1
                                  Test 2
ULAZ:
                                 ULAZ:
0x7F
                                  0x80
IzLAz:
                                 IzLaz:
Najveci:
                                  Najveci:
Naimanii:
                                  Naimanii:
00000000000000000000000001111111
                                  Test 3
                                  Test 4
ULAZ:
                                 ULAZ:
0x00FF00FF
                                  OxFFFFFFFF
TZI.AZ:
                                 TZI.AZ:
Naiveci:
                                  Naiveci:
Najmanji:
                                  Najmanji:
00000000000000011111111111111111
                                  1111111111111111111111111111111111111
```

Zadatak 1.8 Napisati funkcije za rad sa bitovima.

- (a) Napisati funkciju unsigned postavi\_0(unsigned x, unsigned n, unsigned p) koja vraća broj koji se dobija kada se n bitova datog broja x, počevši od pozicije p, postave na 0.
- (b) Napisati funkciju unsigned postavi\_1(unsigned x, unsigned n, unsigned p) koja vraća broj koji se dobija kada se n bitova datog broja x, počevši od pozicije p, postave na 1.
- (c) Napisati funkciju unsigned vrati\_bitove(unsigned x, unsigned n, unsigned p) koja vraća broj u kome se n bitova najmanje težine poklapa sa n bitova broja x počevši od pozicije p, dok su mu ostali bitovi postavljeni na 0.
- (d) Napisati funkciju unsigned postavi\_1\_n\_bitova(unsigned x, unsigned n, unsigned p, unsigned y) koja vraća broj koji se dobija upisivanjem poslednjih n bitova najmanje težine broja y u broj x, počevši od pozicije p.
- (e) Napisati funkciju unsigned invertuj (unsigned x, unsigned n, unsigned p) koja vraća broj koji se dobija invertovanjem n bitova broja x počevši od pozicije p.

Napisati program koji testira prethodno napisane funkcije za neoznačene cele brojeve x, n, p, y koji se unose sa standardnog ulaza. Na standardni izlaz ispisati binarne reprezentacije brojeva x i y, a zatim i binarne reprezentacije brojeva koji se dobijaju pozivanjem prethodno napisanih funkcija. Napomena: Bit najmanje težine je krajnji desni bit i njegova pozicija se označava nultom dok se pozicije ostalih bitova uvećavaju za jedan, sa desna na levo.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite neoznacen ceo broj x: 235
 Unesite neoznacen ceo broj n: 9
 Unesite neoznacen ceo broj p: 24
 Unesite neoznacen ceo broj y: 127
 x = 235
                                              = 00000000000000000000000011101011
                                              = 0000000000000000000000011101011
 postavi_0(
            235,
     235
                                              = 00000000000000000000000011101011
 postavi 1( 235,
                                              = 0000000111111111110000000011101011
       235
                                              = 0000000000000000000000011101011
 vrati_bitove( 235, 9,
                                              24)
       235
                                             = 00000000000000000000000011101011
 у =
      127
                                              = 00000000000000000000000001111111
 postavi_1_n_bitove( 235, 9, 24, 127)
                                             = 000000000111111110000000011101011
                                              = 00000000000000000000000011101011
      235
 invertuj(
            235,
                       24)
                                              = 000000011111111110000000011101011
```

```
INTERAKCIJA SA PROGRAMOM:
 Unesite neoznacen ceo broj x: 2882398951
 Unesite neoznacen ceo broj n: 5
 Unesite neoznacen ceo broj p: 10
 Unesite neoznacen ceo broj y: 35156526
 x = 2882398951
                                                = 101010111100110111101010111100111
 postavi_0(2882398951,
                                                = 10101011110011011110100000100111
                         5, 10)
 x = 2882398951
                                                = 101010111100110111101010111100111
 postavi_1(2882398951, 5, 10)
                                                = 10101011111001101111101111111100111
 x = 2882398951
                                                = 101010111100110111101010111100111
 vrati_bitove(2882398951, 5, 10)
                                                = 000000000000000000000000000001011
 x = 2882398951
                                                = 1010101111100110111101010111100111
 y = 35156526
                                                = 00000010000110000111001000101110
 postavi_1_n_bitove(2882398951, 5, 10, 35156526) = 10101011111001101111101011110101111
 x = 2882398951
                                                = 101010111100110111101010111100111
 invertuj(2882398951,
                      5, 10)
                                                = 101010111110011011110110100100111
```

Zadatak 1.9 Pod rotiranjem bitova ulevo podrazumeva se pomeranje svih bitova za jednu poziciju ulevo, s tim što se bit sa pozicije najveće težine pomera

na poziciju najmanje težine. Analogno, rotiranje bitova udesno podrazumeva pomeranje svih bitova za jednu poziciju udesno, s tim što se bit sa pozicije najmanje težine pomera na poziciju najveće težine.

- (a) Napisati funkciju unsigned rotiraj\_ulevo(unsigned x, unsigned n) koja vraća broj koji se dobija rotiranjem n puta ulevo datog celog neoznačenog broja x.
- (b) Napisati funkciju unsigned rotiraj\_udesno(unsigned x, unsigned n) koja vraća broj koji se dobija rotiranjem n puta udesno datog celog neoznačenog broja x.
- (c) Napisati funkciju int rotiraj\_udesno\_oznaceni(int x, unsigned n) koja vraća broj koji se dobija rotiranjem n puta udesno datog celog broja x.

Napisati program koji sa standardnog ulaza učitava neoznačene cele brojeve  $\mathbf{x}$  i  $\mathbf{n}$  koji se unose u heksadekasnom formatu, tatim ispisuje binarnu reprezentaciju vrednosti dobijene pozivanjem tri prethodno napisane funkcije sa argumentima  $\mathbf{x}$  i  $\mathbf{n}$ , a na kraju ispisuje binarnu reprezentaciju vrednosti dobijene pozivanjem funkcije  $\mathbf{rotiraj}$ \_udesno\_oznaceni za argumente  $-\mathbf{x}$  i  $\mathbf{n}$ .

#### Primer 1

```
| INTERAKCIJA SA PROGRAMOM:
    Unesite neoznacen ceo broj x: balla7
    Unesite neoznacen ceo broj n: 5
    x = 000000010110100001000110100111
    rotiraj_ulevo(balla7, 5) = 00111000000001011101000010001101
    rotiraj_udesno(balla7, 5) = 00111000000001011101000010001101
    rotiraj_udesno_oznaceni(balla7, 5) = 1001111010000000101110100010001101
    rotiraj_udesno_oznaceni(-balla7, 5) = 11000111111111010001011110110010
```

Zadatak 1.10 Napisati funkciju unsigned ogledalo(unsigned x) koja vraća ceo broj čiji binarni zapis predstavlja sliku u ogledalu binarnog zapisa broja x. Napisati program koji testira datu funkciju za broj koji se sa standardnog ulaza zadaje u heksadekadnom formatu. Najpre ispisati binarnu reprezentaciju unetog broja, a zatim i binarnu reprezentaciju broja dobijenog kao njegova slika u ogledalu.

Zadatak 1.11 Napisati funkciju int broj\_01(unsigned int n) koja za dati broj n vraća 1 ako u njegovom binarnom zapisu ima više jednica nego nula, a inače vraća 0. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1	Test 2	Test 3
ULAZ:	ULAZ: 2147377146	ULAZ:
IzLAZ:	IZLAZ:	IZLAZ:

Zadatak 1.12 Napisati funkciju int broj\_parova(unsigned int x) koja vraća broj pojava dve uzastopne jedinice u binarnom zapisu celog neoznačenog broja x. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza. Napomena: *Tri uzastopne jedinice sadrže dve uzastopne jedinice dva puta*.

Test 1	Test 2	Test 3
ULAZ:	ULAZ: 1024	ULAZ: 2147377146
IZLAZ:	IZLAZ:	IZLAZ:

\* Zadatak 1.13 Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na pozicijama i i j. Pozicije i i j učitati kao parametre komandne linije. Pri rešavanju nije dozvoljeno koristiti ni pomoćni niz ni aritmetičke operatore +, -, /, \*, %.

Primer 1	Primer 2	Primer 2
POKRETANJE: ./a.out 1 2	POKRETANJE: ./a.out 1 2	POKRETANJE: ./a.out 12 12
INTERAKCIJA SA PROGRAMOM: ULAZ: 11 IZLAZ: 13	INTERAKCIJA SA PROGRAMOM: ULAZ: 1024 IZLAZ: 1024	INTERAKCIJA SA PROGRAMOM: ULAZ: 12345 IZLAZ: 12345

\* Zadatak 1.14 Napisati funkciju void prevod(unsigned int x, char s[]) koja na osnovu neoznačenog broja x formira nisku s koja sadrži heksadekadni zapis broja x koristeći algoritam za brzo prevođenje binarnog u heksadekadni zapis (svake 4 binarne cifre se zamenjuju jednom odgovarajućom heksade

kadnom cifrom). Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1	Test 2	Test 3
ULAZ:	ULAZ:    1024	ULAZ: 12345
IzLAZ: 0000000B	IZLAZ: 00000400	IzLAZ: 00003039

\* Zadatak 1.15 Napisati funkciju koja za data dva neoznačena broja x i y invertuje one bitove u broju x koji se poklapaju sa odgovarajućim bitovima u broju y. Ostali bitovi treba da ostanu nepromenjeni. Napisati program koji testira tu funkciju za brojeve koji se zadaju sa standardnog ulaza.

Test 1	Test 2	Test 3
ULAZ:	ULAZ:	ULAZ:
123 10	3251 0	12541 1024
IzLAZ:	IZLAZ:	IzLAZ:
4294967285	4294967295	4294966271

Zadatak 1.16 Napisati funkciju koja vraća broj petica u oktalnom zapisu neoznačenog celog broja x. Napisati program koji testira tu funkciju za broj koji se zadaje sa standardnog ulaza. Napomena: Zadatak rešiti isključivo korišćenjem bitskih operatora.

Test 1	Test 2	Test 3
ULAZ: 123	ULAZ: 3245	ULAZ: 100328
IzLAZ:	IZLAZ:	IzLAZ:
0	2	1

## 1.3 Rekurzija

 ${\bf Zadatak~1.17~}$  Napisati rekurzivnu funkciju koja izračunava  $x^k,$  za dati ceo broj xi prirodan broj k

- (a) tako da rešenje bude linearne složenosti,
- (b) tako da rešenje bude logaritamske složenosti.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati redni broj funkcije koju treba primeniti ('1' ili '2'), ceo broj x i prirodan broj k,

a zatim na standarni izlaz ispisati rezultat primene izabrane funkcije na unete brojeve. Ukoliko se na ulazu unese pogrešan redni broj funkcije, ispisati odgovarajuću poruku o grešci na standardni izlaz i prekinuti izvršavanje programa.

Zadatak 1.18 Koristeći uzajamnu (posrednu) rekurziju napisati:

- (a) funkciju unsigned paran(unsigned n) koja proverava da li je broj cifara broja x paran i vraća 1 ako jeste, a 0 inače;
- (b) i funkciju unsigned neparan(unsigned n) koja proverava da li je broj cifara broja x neparan i vraća 1 ako jeste, a 0 inače.

Napisati program koji testira napisane funkcije tako što za heksadekadni broj koji se unosi sa standardnog ulaza ispisuje da li je broj njegovih cifara paran ili neparan.

Zadatak 1.19 Napisati repno-rekurzivnu funkciju koja izračunava faktorijel broja n. Napisati program koji testira napisanu funkciju za proizvoljan broj n ( $n \leq 12$ ) unet sa standardnog ulaza. Napomena: Gornja vrednost za n je postavljena na 12 zbog ograničenja veličine broja koji može da stane u promenljivu tipa int i činjenice da niz faktorijela brzo raste.

```
        Primer 1
        Primer 2

        Interakcija sa programom:
        Interakcija sa programom:

        Unesite n (<= 12): 5</td>
        Unesite n (<= 12): 0</td>

        5! = 120
        0! = 1
```

**Zadatak 1.20** Napisati funkciju koja vraća n-ti element u nizu Fibonačijevih brojeva. Elementi niza Fibonačijevih brojeva F izračunavaju se na osnovu

sledećih rekurentnih relacija:

$$F(0) = 0$$
  
 $F(1) = 1$   
 $F(n) = F(n-1) + F(n-2)$ 

Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati prirodan broj n i na standardni izlaz ispisati rezultat primene napisane funkcije na prirodan broj n.

```
Primer 1

| Interakcija sa programom:
| Unesite koji clan niza se racuna: 5
| F(5) = 5
| F(8) = 21
| Primer 2
| Interakcija sa programom:
| Unesite koji clan niza se racuna: 8
| F(8) = 21
```

**Zadatak 1.21** Elementi niza F izračunavaju se na osnovu sledećih rekurentnih relacija:

$$F(0) = 0$$
  
 $F(1) = 1$   
 $F(n) = a \cdot F(n-1) + b \cdot F(n-2)$ 

Napisati funkciju koja računa n-ti element u nizu F

- (a) iterativno,
- (b) tako da funkcija bude rekurzivna i da koristi navedene rekurentne relacije,
- (c) tako da funkcija bude rekurzivna ali da se problemi manje dimenzije rešavaju samo jedan put.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati redni broj funkcije koju treba primeniti ('1','2','3'), vrednosti koeficijenata a i b i prirodan broj n. Na standardni izlaz ispisati rezultat primene odabrane funkcije nad učitanim podacima, a u slučaju unosa pogrešnog rednog broja funkcije ispisati odgovarajuću poruku i prekinuti izvršavanje pograma. Napomena: Niz F definisan na ovaj način predstavlja uopštenje Fibonačijevih brojeva.

```
Primer 1
                                                  Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                 INTERAKCIJA SA PROGRAMOM:
 Unesite redni broj funkcije:
                                                  Unesite redni broj funkcije:
 1 - iterativna
                                                  1 - iterativna
 2 - rekurzivna
                                                  2 - rekurzivna
 3 - rekurzivna napredna
                                                  3 - rekurzivna napredna
 Unesite koeficijente: 23
                                                  Unesite koeficijente: 4 2
 Unesite koji clan niza se racuna:
                                                  Unesite koji clan niza se racuna: 8
F(5) = 61
                                                  F(8) = 31360
```

Zadatak 1.22 Napisati rekurzivnu funkciju koja sabira dekadne cifre datog celog broja x. Napisati program koji testira ovu funkciju za broj koji se unosi sa standardnog ulaza.



Zadatak 1.23 Napisati rekurzivnu funkciju koja sumira elemente niza celih brojeva

- (a) sabirajući elemente počev od početka niza ka kraju niza,
- (b) sabirajući elemente počev od kraja niza ka početku niza.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati redni broj funkcije ('1' ili '2'), zatim dimenziju  $n \ (0 < n \le 100)$  celobrojnog niza, a potom i elemente niza. Na standardni izlaz ispisati rezultat primene odabrane funkcije nad učitanim nizom, a u slučaju unosa pogrešnog rednog broja funkcije ispisati odgovarajuću poruku i prekinuti izvršavanje pograma.

Zadatak 1.24 Napisati rekurzivnu funkciju koja određuje maksimum niza celih brojeva. Napisati program koji testira ovu funkciju za niz koji se unosi sa standardnog ulaza. Elementi niza se unose sve do kraja ulaza (EOF). Pretpostaviti da niz neće imati više od 256 elemenata.

```
Test 1 Test 2 Test 3

| ULAZ: | ULAZ: | ULAZ: | ULAZ: | ULAZ: | 111 3 5 8 1
| IZLAZ: | IZLAZ: | IZLAZ: | 11
```

Zadatak 1.25 Napisati rekurzivnu funkciju koja izračunava skalarni proizvod dva vektora celih brojeva. Napisati program koji testira ovu funkciju za
nizove (vektore) koji se unose sa standardnog ulaza. Prvo treba uneti dimenziju
nizova, a zatim i njihove elemente. Na standardni izlaz ispisati skalarni proizvod
unetih nizova. Pretpostaviti da nizovi neće imati više od 256 elemenata.

```
Primer 1
                                                  Primer 2
                                                 INTERAKCIJA SA PROGRAMOM:
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju nizova: 3
                                                  Unesite dimenziju nizova: 2
 Unesite elemente prvog niza:
                                                  Unesite elemente prvog niza:
 123
                                                  3 5
 Unesite elemente drugog niza:
                                                  Unesite elemente drugog niza:
 123
                                                  26
 Skalarni proizvod je 14
                                                  Skalarni proizvod je 36
```

**Zadatak 1.26** Napisati rekurzivnu funkciju koja vraća broj pojavljivanja elementa x u nizu a dužine n. Napisati program koji testira ovu funkciju za broj x i niz a koji se unose sa standardnog ulaza. Prvo se unosi x, a zatim elementi niza sve do kraja ulaza. Pretpostaviti da nizovi neće imati više od 256 elemenata.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                 INTERAKCIJA SA PROGRAMOM:
 Unesite ceo broj:
                                                   Unesite ceo broj:
                                                   11
  Unesite elemente niza:
                                                   Unesite elemente niza:
  1234
                                                   3 2 11 14 11 43 1
 Broj pojavljivanja je 1
                                                   Broj pojavljivanja je 2
 Primer 3
INTERAKCIJA SA PROGRAMOM:
 Unesite ceo broj:
 Unesite elemente niza:
  3 21 5 6
 Broj pojavljivanja je 0
```

Zadatak 1.27 Napisati rekurzivnu funkciju kojom se proverava da li su tri data cela broja uzastopni članovi datog celobrojnog niza. Sa standardnog ulaza

učitati tri broja, a zatim elemente niza sve do kraja ulaza. Na standardni izlaz ispisati rezultat primene funkcije nad učitanim podacima. Pretpostaviti da neće biti uneto više od 256 brojeva.

Zadatak 1.28 Napisati rekurzivnu funkciju int prebroj(int x) koja vraća broj bitova postavljenih na 1 u binarnoj reprezentaciji broja x. Napisati program koji testira napisanu funkciju za broj koji se učitava sa standardnog ulaza u heksadekadnom formatu.

Test 1	Test 2	Test 3
ULAZ: 0x7F IZLAZ:	ULAZ:  0x00FF00FF  IZLAZ:	ULAZ:  OxFFFFFFF  IZLAZ:
7	16	32

Zadatak 1.29 Napisati rekurzivnu funkciju koja štampa bitovsku reprezentaciju neoznačenog celog broja, i program koji je testira za vrednost koja se zadaje sa standardnog ulaza.

Zadatak 1.30 Napisati rekurzivnu funkciju za određivanje najveće cifre u oktalnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. UPUT-STVO: Binarne cifre grupisati u podgrupe od po tri cifre, počev od bitova najmanje težine.

Test 1	Test 2	Test 3
ULAZ:	ULAZ:    125	ULAZ:
IZLAZ:	IZLAZ:	IZLAZ:
5	7	1

Zadatak 1.31 Napisati rekurzivnu funkciju za određivanje (dekadne vrednosti) najveće cifre u heksadekadnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. UPUTSTVO: Binarne cifre grupisati u podgrupe od po četiri cifre, počev od bitova najmanje težine.

Test 1	Test 2	Test 3
ULAZ:	ULAZ: 16	ULAZ:
IzLAZ:	IZLAZ:	IZLAZ:
5	1	2

Zadatak 1.32 Napisati rekurzivnu funkciju int palindrom(char s[], int n) koja ispituje da li je data niska s palindrom. Napisati program koji testira ovu funkciju za nisku koja se zadaje sa standardnog ulaza. Pretpostaviti da niska neće imati više od 31 karaktera.

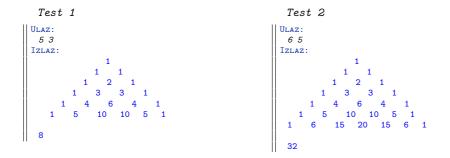


\* Zadatak 1.33 Napisati rekurzivnu funkciju koja prikazuje sve permutacije skupa  $\{1,2,...,n\}$ . Napisati program koji testira napisanu funkciju za proizvoljan prirodan broj  $n\ (n\leq 15)$  unet sa standardnog ulaza.

Test 1	Test 2	Test 3
ULAZ:	ULAZ:	ULAZ:
2   Izlaz:	3 1 2 3	-5   Duzina
1 2	1 3 2	permutacije
2 1	2 1 3	mora biti    broj iz
	3 1 2 3 2 1	intervala [0, 15]!

- \* Zadatak 1.34 Paskalov trougao sadrži brojeve čije se vrednosti računaju tako što svako polje ima vrednost zbira dve vrednosti koje su u susedna dva polja iznad. Izuzetak su jedinice na krajevima. Vrednosti brojeva Paskalovog trougla odgovaraju binomnim koeficijentima tj. vrednost polja (n, k), gde je n redni broj hipotenuze, a k redni broj elementa u tom redu (na toj hipotenuzi) odgovara binomnom koeficijentu  $\binom{n}{k}$ , pri čemu brojanje počinje od nule. Na primer, vrednost polja (4, 2) je 6.
  - (a) Napisati rekurzivnu funkciju koja izračunava vrednost binomnog koeficijenta  $\binom{n}{k}$  koristeći osobine Paskalovog trougla.
  - (b) Napisati rekurzivnu funkciju koja izračunava  $d_n$  kao sumu elemenata n-te hipotenuze Paskalovog trougla.

Napisati program koji za unetu veličinu Paskalovog trougla i redni broj hipotenuze najpre iscrtava Paskalov trougao, a zatim štampa sumu elemenata hipotenuze.



\* Zadatak 1.35 Napisati rekurzivnu funkciju koja prikazuje sve varijacije sa ponavljanjem dužine n skupa  $\{a,b\}$ , i program koji je testira, za n koje se unosi sa standardnog ulaza.

Test 1	Test 2
ULAZ:	ULAZ:
2	3
IZLAZ:	Izlaz:
a a	aaa
a b	aab
b a	aba
b b	a b b
	baa
	bab
	bba
	b b b

\* Zadatak 1.36 Hanojske kule: Data su tri vertikalna štapa. Na jednom od njih se nalazi n diskova poluprečnika 1, 2, 3,... do n, tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala dva štapa su prazna. Potrebno je premestiti diskove sa jednog na drugi štap tako da budu u istom redosledu, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg. Preostali štap koristiti kao pomoćni štap prilikom premeštanja.

Napisati program koji za proizvoljnu vrednost n, koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

\* Zadatak 1.37 Modifikacija Hanojskih kula: Data su četiri vertikalna štapa. Na jednom se nalazi <math>n diskova poluprečnika 1, 2, 3,... do n, tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala tri štapa su prazna. Potrebno je premestiti diskove na drugi štap tako da budu u istom redosledu, premestajući jedan po jedan disk, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg. Preostala dva štapa koristiti kao pomoćne štapove prilikom premeštanja. Napisati program koji za proizvoljnu vrednost n, koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

## 1.4 Rešenja

#### Rešenje 1.1

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

/* Struktura kojom je predstavljan kompleksan broj sadrzi realan i
imaginaran deo kompleksnog broja */
typedef struct {
  float real;
  float imag;
```

```
10 } KompleksanBroj;
  /* Funkcija ucitava sa standardnog ulaza realan i imaginara deo
     kompleksnog broja i smesta ih u strukturu cija je adresa argument
     funkcije */
14
  void ucitaj_kompleksan_broj(KompleksanBroj * z)
  {
16
    /* Ucitava se vrednost sa standardnog ulaza */
    printf("Unesite realni i imaginarni deo kompleksnog broja: ");
18
    scanf("%f", &z->real);
    scanf("%f", &z->imag);
20
  /* Funkcija ispisuje na standardan izlaz zadati kompleksni broj u
     obliku (x + i y). Ovoj funkciji se argument prenosi po vrednosti
24
     jer se u samoj funkciji ne menja njegova vrednost */
  void ispisi_kompleksan_broj(KompleksanBroj z)
26
    /* Zapocinje se sa ispisom */
28
    printf("(");
30
    /* Razlikuju se dva slucaja: 1) realni deo kompleksnog broja
       razlicit od nule: tada se realni deo ispisuje na standardni
       izlaz, nakon cega se ispisuje znak + ili - u zavisnosti da li
       je imaginarni deo pozitivan ili negativan, a potom i apsolutna
34
       vrednost imaginarnog dela kompleksnog broja 2) realni deo
       kompleksnog broja je nula: tada se samo ispisuje imaginaran
36
       deo, s tim sto se ukoliko su oba dela nula ispisuje samo 0, bez
       decimalnih mesta */
38
    if (z.real != 0) {
      printf("%.2f", z.real);
40
      if (z.imag > 0)
42
        printf(" + %.2f i", z.imag);
      else if (z.imag < 0)</pre>
44
        printf(" - %.2f i", -z.imag);
46
    } else {
      if (z.imag == 0)
        printf("0");
48
      else
50
        printf("%.2f i", z.imag);
    /* Zavrsava se sa ispisom */
    printf(")");
54
56
  /* Funkcija vraca vrednosti realnog dela kompleksnog broja */
58 float realan_deo(KompleksanBroj z)
    return z.real;
```

```
/* Funkcija vraca vrednosti imaginarnog dela kompleksnog broja */
64 float imaginaran_deo(KompleksanBroj z)
   return z.imag;
  }
68
   /* Funkcija vraca vrednost modula zadatog kompleksnog broja */
70 float moduo(KompleksanBroj z)
    return sqrt(z.real * z.real + z.imag * z.imag);
74
   /* Funkcija vraca vrednost konjugovano kompleksnog broja koji
     odgovara kompleksnom broju argumentu */
   KompleksanBroj konjugovan(KompleksanBroj z)
78 -
    /* Konjugovano kompleksan broj z se dobija tako sto se promeni
       znak imaginarnom delu kompleksnog broja */
80
    KompleksanBroj z1 = z;
    z1.imag *= -1;
82
    return z1;
84
86
   /* Funkcija vraca kompleksan broj cija vrednost je jednaka zbiru
     argumenata funkcije */
88
   KompleksanBroj saberi(KompleksanBroj z1, KompleksanBroj z2)
90
    /* Rezultat sabiranja dva kompleksna broja z1 i z2 je kompleksan
        broj ciji je realan deo zbir realnih delova kompleksnih brojeva
        z1 i z2, a imaginaran deo zbir imaginarnih delova kompleksnih
       brojeva z1 i z2 */
94
    KompleksanBroj z = z1;
96
    z.real += z2.real;
    z.imag += z2.imag;
98
    return z;
100 }
102 /* Funkcija vraca kompleksan broj cija vrednost je jednaka razlici
      argumenata funkcije */
104 KompleksanBroj oduzmi(KompleksanBroj z1, KompleksanBroj z2)
    /* Rezultat oduzimanja dva kompleksna broja z1 i z2 je kompleksan
106
        broj ciji je realan deo razlika realnih delova kompleksnih
        brojeva z1 i z2, a imaginaran deo razlika imaginarnih delova
108
        kompleksnih brojeva z1 i z2 */
    KompleksanBroj z = z1;
110
    z.real -= z2.real;
    z.imag -= z2.imag;
```

```
return z;
   /* Funkcija vraca kompleksan broj cija vrednost je jednaka proizvodu
      argumenata funkcije */
118
   KompleksanBroj mnozi(KompleksanBroj z1, KompleksanBroj z2)
120
     /* Rezultat mnozenja dva kompleksna broja z1 i z2 je kompleksan
        broj ciji se realan i imaginaran deo racunaju po formuli za
        mnozenje kompleksnih brojeva z1 i z2 */
     KompleksanBroj z;
124
     z.real = z1.real * z2.real - z1.imag * z2.imag;
     z.imag = z1.real * z2.imag + z1.imag * z2.real;
126
     return z;
128
130
   /* Funkcija vraca argument zadatog kompleksnog broja */
  float argument(KompleksanBroj z)
     /* Argument kompleksnog broja z se racuna pozivanjem funkcije
134
        atan2 iz biblioteke math.h */
     return atan2(z.imag, z.real);
136
138
   int main()
   {
140
     char c;
142
     /* Deklaracija 3 promenljive tipa KompleksanBroj */
     KompleksanBroj z1, z2, z;
144
     /* Ucitava se prvi kompleksni broj, koji se potom ispisuje na
146
        standardni izlaz */
148
     ucitaj_kompleksan_broj(&z1);
     ispisi_kompleksan_broj(z1);
     printf("\n");
     /* Ucitava se drugi kompleksni broj, koji se potom ispisuje na
        standardni izlaz */
     ucitaj_kompleksan_broj(&z2);
     ispisi_kompleksan_broj(z2);
     printf("\n");
156
     /* Ucitavase znak na osnovu koga korisnik bira aritmeticku
        operaciju koja ce se izvrsiti nad kompleksnim brojevima, a
        zatim se vrsi provera da li je unet neki od dozvoljenih
160
        aritmetickih znakova. */
     getchar();
     printf("Unesite znak (+,-,*): ");
     scanf("%c", &c);
164
     if (c != '+' && c != '-' && c != '*') {
```

```
fprintf(stderr, "Greska: Nedozvoljena vrednost operatora.\n");
166
       exit(EXIT_FAILURE);
168
     /* Analizira se uneti operator */
     if (c == '+') {
       /* Racuna se zbir */
       z = saberi(z1, z2);
     } else if (c == '-') {
       /* Racuna se razlika */
       z = oduzmi(z1, z2);
     } else {
       /* Racuna se proizvod */
178
       z = mnozi(z1, z2);
180
     /* Ispisuje se rezultat */
182
     ispisi_kompleksan_broj(z1);
     printf(" %c ", c);
184
     ispisi_kompleksan_broj(z2);
     printf(" = ");
186
     ispisi_kompleksan_broj(z);
188
     /* Ispisuje se realan, imaginaran deo i moduo prvog kompleksnog
        broja */
190
     printf("\nRealni_deo: %.f\nImaginarni_deo: %f\nModuo: %f\n",
            realan_deo(z), imaginaran_deo(z), moduo(z));
     /* Izracunava se i ispisuje konjugovano kompleksan broj drugog
194
        kompleksnog broja */
     printf("Konjugovano kompleksan broj: ");
196
     ispisi_kompleksan_broj(konjugovan(z));
     printf("\n");
198
200
     /* Testira se funkcija koja racuna argument kompleksnog broja */
     printf("Argument kompleksnog broja: %f\n", argument(z));
202
     exit(EXIT_SUCCESS);
204
```

#### Rešenje 1.2

#### kompleksan\_broj.h

```
/* Zaglavlje kompleksan_broj.h sadrzi definiciju tipa KompleksanBroj i deklaracije funkcija za rad sa kompleksnim brojevima. Zaglavlje nikada ne treba da sadrzi definicije funckija. Da bi neki program mogao da koristi ove brojeve i funkcije iz ove biblioteke, neophodno je da ukljuci ovo zaglavlje. */
```

```
7 /* Ovim pretprocesorskim direktivama se zakljucava zaglavlje i
     onemogucava se da se sadrzaj zaglavlja vise puta ukljuci. Niska
     posle kljucne reci ifndef je proizvoljna, ali treba da se ponovi
     u narednoj pretrocesorskoj define direktivi. */
11 #ifndef KOMPLEKSAN BROJ H
  #define _KOMPLEKSAN_BROJ_H
13
  /* Zaglavlja standardne biblioteke koje sadrze deklaracije funkcija
     koje se koriste u definicijama funkcija navedenim u
     kompleksan_broj.c */
  #include <stdio.h>
  #include <math.h>
19
  /* Struktura KompleksanBroj */
21 typedef struct {
    float real;
   float imag;
23
  } KompleksanBroj;
  /* Deklaracije funkcija za rad sa kompleksnim brojevima. Sve one su
     definisane u kompleksan_broj.c */
  /* Funkcija ucitava sa standardnog ulaza realan i imaginara deo
29
     kompleksnog broja i smesta ih u strukturu cija je adresa argument
     funkcije */
31
  void ucitaj_kompleksan_broj(KompleksanBroj * z);
33
  /* Funkcija ispisuje na standardan izlaz zadati kompleksni broj u
     obliku (x + i y) */
  void ispisi_kompleksan_broj(KompleksanBroj z);
  /* Funkcija vraca vrednosti realnog dela kompleksnog broja */
39 float realan_deo(KompleksanBroj z);
  /* Funkcija vraca vrednosti imaginarnog dela kompleksnog broja */
  float imaginaran_deo(KompleksanBroj z);
43
  /* Funkcija vraca vrednost modula zadatog kompleksnog broja */
45 float moduo(KompleksanBroj z);
  /* Funkcija vraca vrednost konjugovano kompleksnog broja koji
     odgovara kompleksnom broju argumentu */
49 KompleksanBroj konjugovan(KompleksanBroj z);
  /* Funkcija vraca kompleksan broj cija vrednost je jednaka zbiru
     argumenata funkcije */
KompleksanBroj saberi(KompleksanBroj z1, KompleksanBroj z2);
55 /* Funkcija vraca kompleksan broj cija vrednost je jednaka razlici
     argumenata funkcije */
57 KompleksanBroj oduzmi (KompleksanBroj z1, KompleksanBroj z2);
```

```
/* Funkcija vraca kompleksan broj cija vrednost je jednaka proizvodu
argumenata funkcije */
KompleksanBroj mnozi(KompleksanBroj z1, KompleksanBroj z2);

/* Funkcija vraca argument zadatog kompleksnog broja */
float argument(KompleksanBroj z);

/* Kraj zakljucanog dela */
#endif
```

#### kompleksan\_broj.c

```
1 /* Ukljucuje se zaglavlje za rad sa kompleksnim brojevima, jer je
     neophodno da bude poznata definicija tipa KompleksanBroj.
     Takodje, time su ukljucena zaglavlja standardne biblioteke koja
     su navedena u kompleksan_broj.h */
5 #include "kompleksan_broj.h"
void ucitaj_kompleksan_broj(KompleksanBroj * z)
    /* Ucitavanje vrednosti sa standardnog ulaza */
   printf("Unesite realan i imaginaran deo kompleksnog broja: ");
   scanf("%f", &z->real);
    scanf("%f", &z->imag);
13 }
void ispisi_kompleksan_broj(KompleksanBroj z)
    /* Zapocinje se sa ispisom */
17
    printf("(");
19
    /* Razlikuju se dva slucaja: 1) realni deo kompleksnog broja
       razlicit od nule: tada se realni deo ispisuje na standardni
       izlaz, nakon cega se ispisuje znak + ili - u zavisnosti da li
       je imaginarni deo pozitivan ili negativan, a potom i apsolutna
       vrednost imaginarnog dela kompleksnog broja 2) realni deo
       kompleksnog broja je nula: tada se samo ispisuje imaginaran
       deo, s tim sto se ukoliko su oba dela nula ispisuje samo 0, bez
       decimalnih mesta */
27
    if (z.real != 0) {
29
      printf("%.2f", z.real);
      if (z.imag > 0)
        printf(" + %.2f i", z.imag);
      else if (z.imag < 0)
        printf(" - %.2f i", -z.imag);
    } else {
35
      if (z.imag == 0)
        printf("0");
        printf("%.2f i", z.imag);
39
```

```
/* Zavrsava se sa ispisom */
    printf(")");
45
  float realan_deo(KompleksanBroj z)
47
    /* Vraca se vrednost realnog dela kompleksnog broja */
    return z.real;
49
  float imaginaran_deo(KompleksanBroj z)
    /* Vraca se vrednost imaginarnog dela kompleksnog broja */
    return z.imag;
  float moduo(KompleksanBroj z)
59
    /* Koriscenjem funkcije sqrt racuna se moduo kompleksnog broja */
    return sqrt(z.real * z.real + z.imag * z.imag);
61
63
  KompleksanBroj konjugovan(KompleksanBroj z)
65
    /* Konjugovano kompleksan broj se dobija od datog broja z tako sto
       se promeni znak imaginarnom delu kompleksnog broja */
67
    KompleksanBroj z1 = z;
    z1.imag *= -1;
69
    return z1;
  }
71
  KompleksanBroj saberi(KompleksanBroj z1, KompleksanBroj z2)
    /* Rezultat sabiranja dva kompleksna broja z1 i z2 je kompleksan
       broj ciji je realan deo zbir realnih delova kompleksnih brojeva
       z1 i z2, a imaginaran deo zbir imaginarnih delova kompleksnih
       brojeva z1 i z2 */
    KompleksanBroj z = z1;
79
    z.real += z2.real;
    z.imag += z2.imag;
81
83
    return z;
85
  KompleksanBroj oduzmi(KompleksanBroj z1, KompleksanBroj z2)
87
    /* Rezultat oduzimanja dva kompleksna broja z1 i z2 je kompleksan
       broj ciji je realan deo razlika realnih delova kompleksnih
89
       brojeva z1 i z2, a imaginaran deo razlika imaginarnih delova
       kompleksnih brojeva z1 i z2 */
```

```
KompleksanBroj z = z1;
     z.real -= z2.real;
93
     z.imag -= z2.imag;
95
     return z:
97 }
99 KompleksanBroj mnozi(KompleksanBroj z1, KompleksanBroj z2)
     /* Rezultat mnozenja dva kompleksna broja z1 i z2 je kompleksan
        broj ciji se realan i imaginaran deo racunaju po formuli za
        mnozenje kompleksnih brojeva z1 i z2 */
     KompleksanBroj z;
     z.real = z1.real * z2.real - z1.imag * z2.imag;
    z.imag = z1.real * z2.imag + z1.imag * z2.real;
     return z;
109 }
111 float argument (KompleksanBroj z)
     /* Argument kompleksnog broja z se racuna pozivanjem funkcije
113
        atan2 iz biblioteke math.h */
    return atan2(z.imag, z.real);
```

#### main.c

```
2 Ovaj program koristi korektno definisanu biblioteku kompleksnih
  brojeva. U zaglavlju kompleksan_broj.h nalazi se definicija
4 komplesnog broja i popis deklaracija podrzanih funkcija, a u
  kompleksan_broj.c se nalaze njihove definicije.
  Kompilacija programa se najjednostavnije postize naredbom
s gcc -Wall -lm -o kompleksan_broj kompleksan_broj.c main.c
10 Kompilacija se moze uraditi i na sledeci nacin:
  gcc -Wall -c -o kompleksan_broj.o kompleksan_broj.c
12 gcc -Wall -c -o main.o main.c
  gcc -lm -o kompleksan_broj kompleksan_broj.o main.o
14
  Napomena: Prethodne komande se koriste kada se sva tri navedena
16 dokumenta nalaze u istom direktorijumu. Ukoliko se biblioteka (npr.
  kompleksan_broj.c kompleksan_broj.h) nalazi u direktorijumu sa imenom
18 header_dir prevodjenje se vrsi dodavanjem opcije opcije -I header_dir
  gcc -I header_dir -Wall -lm -o kompleksan_broj kompleksan_broj.c
20
      main.c
```

```
24 #include <stdio.h>
  /* Ukljucuje se zaglavlje neophodno za rad sa kompleksnim brojevima
26 #include "kompleksan_broj.h"
  /* U glavnoj funkciji se za uneti kompleksan broj ispisuje njegov
     polarni oblik */
  int main()
30
    KompleksanBroj z;
32
    /* Ucitava se kompleksan broj */
34
    ucitaj_kompleksan_broj(&z);
36
    /* Ispisuje se polarni oblik kompleksnog broja */
    printf("Polarni oblik kompleksnog broja je %.2f * e^i * %.2f\n",
38
           moduo(z), argument(z));
40
    return 0;
42 }
```

#### Rešenje 1.3

#### polinom.h

```
1 #ifndef _POLINOM_H
  #define _POLINOM_H
  #include <stdio.h>
5 #include <stdlib.h>
  /* Maksimalni stepen polinoma */
  #define MAKS_STEPEN 20
  /* Polinomi se predstavljaju strukturom koja cuva koeficijente
     (koef[i] je koeficijent uz clan x^i) i stepen polinoma */
13 typedef struct {
    double koef[MAKS_STEPEN + 1];
    int stepen;
  } Polinom;
  /* Funkcija koja ispisuje polinom na standardni izlaz u citljivom
     obliku. Polinom se prenosi po adresi da bi se ustedela memorija:
     ne kopira se cela struktura, vec se samo prenosi adresa na kojoj
     se nalazi polinom koji ispisujemo */
  void ispisi(const Polinom * p);
  /* Funkcija koja ucitava polinom sa tastature */
25 Polinom ucitaj();
```

```
/* Funkcija racuna i vraca vrednost polinoma p u tacki x Hornerovim algoritmom */
double izracunaj(const Polinom * p, double x);

/* Funkcija koja sabira dva polinoma */
Polinom saberi(const Polinom * p, const Polinom * q);

/* Funkcija koja mnozi dva polinoma p i q */
Polinom pomnozi(const Polinom * p, const Polinom * q);

/* Funkcija koja racuna izvod polinoma p */
Polinom izvod(const Polinom * p);

/* Funkcija koja racuna n-ti izvod polinoma p */
Polinom n_izvod(const Polinom * p, int n);
#endif
```

#### polinom.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "polinom.h"
  void ispisi(const Polinom * p)
6
  {
    int nulaPolinom = 1;
    /* Ispisivanje polinoma pocinje od najviseg stepena ka najnizem da
       bi polinom bio ispisan na prirodan nacin. Ipisisuju se samo oni
       koeficijenti koji su razliciti od nule. Ispred pozitivnih
       koeficijenata je potrebno ispisati znak + (osim u slucaju
12
       koeficijenta uz najvisi stepen). */
    for (i = p->stepen; i >= 0; i--) {
14
16
      if (p->koef[i]) {
        /* Polinom nije nula polinom, cim je neki od koeficijenata
18
           razlicit od nule */
        nulaPolinom = 0;
20
        if (p->koef[i] >= 0 && i != p->stepen)
          putchar('+');
        if (i > 1)
          printf("%.2fx^%d", p->koef[i], i);
        else if (i == 1)
24
          printf("%.2fx", p->koef[i]);
26
          printf("%.2f", p->koef[i]);
      }
28
    }
    /* U slucaju nula polinoma indikator ce imati vrednost 1 i tada se
30
       ispisuje nula. */
```

```
if (nulaPolinom)
32
      printf("0");
    putchar('\n');
34
36
  Polinom ucitaj()
  {
38
    int i:
    Polinom p;
40
    /* Ucitava se stepena polinoma */
42
    scanf("%d", &p.stepen);
44
    /* Ponavlja se ucitavanje stepena sve dok se ne unese stepen iz
       dozvoljenog opsega */
46
    while (p.stepen > MAKS_STEPEN || p.stepen < 0) {
      printf("Stepen polinoma pogresno unet, pokusajte ponovo: ");
48
      scanf("%d", &p.stepen);
    /* Unose se koeficijenti polinoma */
    for (i = p.stepen; i >= 0; i--)
      scanf("%lf", &p.koef[i]);
54
    /* Vraca se procitani polinom */
56
    return p;
  }
58
  double izracunaj(const Polinom * p, double x)
60
    /* Rezultat se na pocetku inicijalizuje na nulu, a potom se u
       svakoj iteraciji najpre mnozi sa x, a potom i uvecava za
       vrednost odgovarajuceg koeficijenta */
64
    /* Primer: Hornerov algoritam za polinom x^4+2x^3+3x^2+2x+1:
       x^4+2x^3+3x^2+2x+1 = (((x+2)*x + 3)*x + 2)*x + 1 */
68
    double rezultat = 0;
    int i = p->stepen;
70
    for (; i >= 0; i--)
      rezultat = rezultat * x + p->koef[i];
    return rezultat;
  }
74
76 Polinom saberi(const Polinom * p, const Polinom * q)
    Polinom rez;
78
    int i;
80
    /* Stepen rezultata ce odgovarati stepenu polinoma sa vecim
       stepenom */
82
    rez.stepen = p->stepen > q->stepen ? p->stepen : q->stepen;
```

```
84
     /* Racunaju se svi koeficijenti rezultujuceg polinoma tako sto se
        sabiraju koeficijenti na odgovarajucim pozicijama polinoma koje
86
        sabiramo. Ukoliko je pozicija za koju se racuna koeficijent veca
        od stepena nekog od polaznih polinoma podrazumeva se da je
88
        koeficijent jednak koeficijentu uz odgovarajuci stepen iz drugog
        polinoma */
90
     for (i = 0; i <= rez.stepen; i++)
       rez.koef[i] =
           (i > p->stepen ? 0 : p->koef[i]) +
           (i > q->stepen ? 0 : q->koef[i]);
94
     /* Vraca se dobijeni polinom */
96
     return rez;
98
Polinom pomnozi(const Polinom * p, const Polinom * q)
     int i, j;
     Polinom r;
104
     /* Stepen rezultata ce odgovarati zbiru stepena polaznih polinoma
     r.stepen = p->stepen + q->stepen;
106
     if (r.stepen > MAKS_STEPEN) {
      fprintf(stderr, "Stepen proizvoda polinoma izlazi iz opsega\n");
108
       exit(EXIT_FAILURE);
     }
     /* Svi koeficijenti rezultujuceg polinoma se inicijalizuju na 0 */
     for (i = 0; i <= r.stepen; i++)
      r.koef[i] = 0;
114
     /* U svakoj iteraciji odgovarajuci koeficijent rezultata se
        uvecava za proizvod odgovarajucih koeficijenata iz polaznih
        polinoma */
118
     for (i = 0; i <= p->stepen; i++)
       for (j = 0; j \le q -> stepen; j++)
         r.koef[i + j] += p->koef[i] * q->koef[j];
     /* Vraca se dobijeni polinom */
     return r;
124
   Polinom izvod(const Polinom * p)
128
     int i;
     Polinom r;
130
     /* Izvod polinoma ce imati stepen za jedan stepen manji od stepena
        polaznog polinoma. Ukoliko je stepen polinoma p vec nula, onda
        je rezultujuci polinom nula (izvod od konstante je nula). */
134
```

```
if (p->stepen > 0) {
       r.stepen = p->stepen - 1;
136
       /* Racunanje koeficijenata rezultata na osnovu koeficijenata
138
          polaznog polinoma */
       for (i = 0; i <= r.stepen; i++)
140
         r.koef[i] = (i + 1) * p->koef[i + 1];
     } else
142
       r.koef[0] = r.stepen = 0;
144
     /* Vraca se dobijeni polinom */
146
     return r;
148
   Polinom n_izvod(const Polinom * p, int n)
     int i;
     Polinom r;
     /* Provera da li je n nenegativna vrednost */
154
     if (n < 0) {
       fprintf(stderr, "U n-tom izvodu polinoma, n mora biti >=0 \n");
156
       exit(EXIT_FAILURE);
158
     /* Nulti izvod je bas taj polinom */
160
     if (n == 0)
       return *p;
     /* Za n>=1, n-ti izvod se racuna tako sto se n puta pozove
164
        funkcija za racunanje prvog izvoda polinoma */
     r = izvod(p);
166
     for (i = 1; i < n; i++)
       r = izvod(&r);
168
     /* Vraca se dobijeni polinom */
     return r;
172 }
```

#### main.c

```
#include <stdio.h>
#include "polinom.h"

int main(int argc, char **argv)
{
    Polinom p, q, z, r;
    double x;
    int n;

/* Unosi se polinom p */
```

```
printf
        ("Unesite polinom p (prvo stepen, pa zatim koeficijente od
12
      najveceg stepena do nultog):\n");
    p = ucitaj();
14
    /* Ispisuje se polinom p */
    ispisi(&p);
    /* Unosi se polinom q */
18
    printf
        ("Unesite drugi polinom q (prvo stepen, pa zatim koeficijente
20
      od najveceg stepena do nultog):\n");
    q = ucitaj();
    /* Polinomi se sabiraju i ispisuje se izracunati zbir */
    z = saberi(&p, &q);
24
    printf("Zbir polinoma je polinom z:\n");
    ispisi(&z);
26
    /* Polinomi se mnoze i ispisuje se izracunati prozivod */
28
    r = pomnozi(&p, &q);
    printf("Prozvod polinoma je polinom r:\n");
30
    ispisi(&r);
    /* Ispisuje se vrednost polinoma u unetoj tacki */
    printf("Unesite tacku u kojoj racunate vrednost polinoma z:\n");
34
    scanf("%lf", &x);
    printf("Vrednost polinoma z u tacki %.2f je %.2f\n", x,
36
           izracunaj(&z, x));
38
    /* Racuna se n-ti izvoda polinoma i ispisuje se dobijeni polinoma
    printf("Unesite izvod polinoma koji zelite:\n");
40
    scanf("%d", &n);
42
    r = n_{izvod(\&r, n)};
    printf("%d. izvod polinoma r je: ", n);
44
    ispisi(&r);
    exit(EXIT_SUCCESS);
46
```

 $stampanje\_bitova.h$ 

```
#ifndef _STAMPANJE_BITOVA_H
#define _STAMPANJE_BITOVA_H

#include <stdio.h>
```

```
/* Funkcija prikazuje na standardni izlaz binarnu reprezentaciju
celog broja u memoriji. Bitove koji predstavljaju binarnu
reprezentaciju broja treba ispisati sa leva na desno, tj. od bita
najvece tezine ka bitu najmanje tezine */
void stampaj_bitove(unsigned x);

/* Funkcija prikazuje na standardni izlaz binarnu reprezentaciju
celog broja tipa 'short' u memoriji. */
void stampaj_bitove_short(short x);

/* Funkcija prikazuje na standardni izlaz binarnu reprezentaciju
karaktera u memoriji. */
void stampaj_bitove_char(char x);

#endif
```

#### stampanje\_bitova.c

```
#include <stdio.h>
# #include "stampanje_bitova.h"
  void stampaj_bitove(unsigned x)
    /* Broj bitova celog broja */
    unsigned velicina = sizeof(unsigned) * 8;
    /* Maska koja se koristi za "ocitavanje" bitova celog broja */
    unsigned maska;
    /* Pocetna vrednost maske se postavlja na broj ciji binarni zapis
       na mestu bita najvece tezine sadrzi jedinicu, a na svim ostalim
       mestima sadrzi nulu. U svakoj iteraciji maska se menja tako sto
       se jedini bit jedinica pomera udesno, kako bi se odredio naredni
       bit broja x koji je argument funkcije. Zatim se odgovarajuca
       cifra, ('0' ili '1'), ispisuje na standardnom izlazu. Neophodno
18
       je da promenljiva maska bude deklarisana kao neoznacen ceo broj
       kako bi se pomeranjem u desno vrsilo logicko pomeranje
       (popunjavanje nulama), a ne aritmeticko pomeranje (popunjavanje
       znakom broja). */
    for (maska = 1 << (velicina - 1); maska != 0; maska >>= 1)
      putchar(x & maska ? '1' : '0');
    putchar('\n');
26 }
void stampaj_bitove_short(short x)
    /* Broj bitova celog broja tipa short */
    unsigned velicina = sizeof(short) * 8;
    /* Maska koja se koristi za "ocitavanje" bitova broja tipa short */
```

```
unsigned short maska;
    for (maska = 1 << (velicina - 1); maska != 0; maska >>= 1)
36
      putchar(x & maska ? '1' : '0');
38
    putchar('\n');
  }
40
  |void stampaj_bitove_char(char x)
42
    /* Broj bitova karaktera */
44
    unsigned velicina = sizeof(char) * 8;
46
    /* Maska koja se koristi za "ocitavanje" bitova jednog karaktera */
    unsigned char maska;
48
   for (maska = 1 << (velicina - 1); maska != 0; maska >>= 1)
      putchar(x & maska ? '1' : '0');
    putchar('\n');
54 }
```

main.c

```
#include <stdio.h>
#include "stampanje_bitova.h"
4 int main()
  {
    int broj_int;
6
    short broj_short;
    char broj_char;
    /* Ucitava se broj tipa int */
10
    printf("Unesite broj tipa int: ");
    scanf("%x", &broj_int);
14
    /* Ispisuje se binarna reprezentacija unetog broja */
    printf("Binarna reprezentacija: ");
16
    stampaj_bitove(broj_int);
    /* Ucitava se broj tipa short */
18
    printf("Unesite broj tipa short: ");
20
    scanf("%hx", &broj_short);
    /* Ispisuje se binarna reprezentacija unetog broja */
    printf("Binarna reprezentacija: ");
    stampaj_bitove_short(broj_short);
24
    /* Ucitava se broj tipa char */
26
    printf("Unesite broj tipa char: ");
```

```
scanf("%hhx", &broj_char);

/* Ispisuje se binarna reprezentacija unetog broja */
printf("Binarna reprezentacija: ");
stampaj_bitove_char(broj_char);

return 0;
}
```

```
#include <stdio.h>
  #include <stdlib.h>
  /* Funkcija vraca broj jedinica u binarnoj reprezentaciji broja x
     kreiranjem odgovarajuce maske i njenim pomeranjem */
  int prebroj_bitove_1(int x)
    int br = 0;
    unsigned broj_pomeranja = sizeof(unsigned) * 8 - 1;
    /* Formiranje se maska cija binarna reprezentacija izgleda
       100000...0000000, koja sluzi za ocitavanje bita najvece tezine.
       U svakoj iteraciji maska se pomera u desno za 1 mesto, i ocitava
       se sledeci bit. Petlja se zavrsava kada vise nema jedinica tj.
       kada maska postane nula. */
    unsigned maska = 1 << broj_pomeranja;
    for (; maska != 0; maska >>= 1)
      x & maska ? br++ : 1;
    return br;
21 }
  /* Funkcija vraca broj jedinica u binarnoj reprezentaciji broja x
     formiranjem odgovarajuce maske i pomeranjem promenljive x */
  int prebroj_bitove_2(int x)
    int br = 0;
    unsigned broj_pomeranja = sizeof(int) * 8 - 1;
    /* Kako je argument funkcije oznacen ceo broj x naredba x>>=1 bi
       vrsila aritmeticko pomeranje u desno, tj. popunjavanje bita
       najvece tezine bitom znaka. U tom slucaju nikad ne bi bio
       ispunjen uslov x!=0 i program bi bio zarobljen u beskonacnoj
       petlji. Zbog toga se koristi pomeranje broja x ulevo i maska
       koja ocitava bit najvece tezine. */
35
    unsigned maska = 1 << broj_pomeranja;
    for (; x != 0; x <<= 1)
      x & maska ? br++ : 1;
39
```

```
return br;
43
  int main()
45 | {
    int x, i;
47
    /* Ucitava se broj sa ulaza */
    printf("Unesite broj:\n");
    scanf("%x", &x);
    /* Dozvoljava se korisniku da bira na koji nacin ce biti izracunat
       broj jedinica u zapisu broja */
    printf("Unesite redni broj funkcije:\n");
    scanf("%d", &i);
    /* Ispisuje se odgovarajuci rezultat na osnovu unesenog rednog
57
       broja funkcije */
    if (i == 1) {
      printf("Poziva se funkcija prebroj_bitove_1\n");
      printf("Broj jedinica u zapisu je %d\n", prebroj_bitove_1(x));
    } else if (i == 2) {
      printf("Poziva se funkcija prebroj_bitove_2\n");
      printf("Broj jedinica u zapisu je %d\n", prebroj_bitove_2(x));
    } else {
      fprintf(stderr, "Greska: Neodgovarajuci redni broj funkcije.\n");
      exit(EXIT_FAILURE);
    exit(EXIT_SUCCESS);
  }
```

```
#include <stdio.h>
#include "stampanje_bitova.h"

/* Funkcija vraca najveci neoznaceni broj sastavljen od istih bitova
    koji se nalaze u binarnoj reprezentaciji vrednosti promenjive x */
unsigned najveci(unsigned x)
{
    unsigned velicina = sizeof(unsigned) * 8;

/* Formira se maska 100000...0000000 */
unsigned maska = 1 << (velicina - 1);

/* Rezultat se inicijalizuje vrednoscu 0 */
unsigned rezultat = 0;</pre>
```

```
/* Promenljiva x se pomera u levo sve dok postoje jedinice u
       njenoj binarnoj reprezentaciji (tj. sve dok je promenljiva x
17
       razlicita od nule). */
    for (; x != 0; x <<= 1) {
19
      /* Za svaku jedinicu koja se koriscenjem maske detektuje na
         poziciji najvece tezine u binarnoj reprezentaciji promenjive
         x, potiskuje se jedna nova jedinicu sa leva u rezultat */
      if (x & maska) {
        rezultat >>= 1;
        rezultat |= maska;
    /* Vraca se dobijena vrednost */
    return rezultat;
  }
31
  /* Funkcija vraca najmanji neoznaceni broj sastavljen od istih
     bitova koji se nalaze u binarnoj reprezentaciji vrednosti
     promenjive x */
  unsigned najmanji (unsigned x)
37
    /* Rezultat se inicijalizuje vrednoscu 0 */
    unsigned rezultat = 0;
39
    /* Promenljiva x se pomera u desno sve dok postoje jedinice u
41
       njenoj binarnoj reprezentaciji (tj. sve dok je promenljiva x
       razlicita od nule). */
43
    for (; x != 0; x >>= 1) {
      /* Za svaku jedinicu koja se koriscenjem vrednosti 1 za masku
45
         detektuje na poziciji najmanje tezine u binarnoj
         reprezentaciji promenjive x, potiskuje se jedna nova jedinicu
47
         sa desna u rezultat */
      if (x & 1) {
49
        rezultat <<= 1;
        rezultat |= 1;
      }
53
    /* Vraca se dobijena vrednost */
    return rezultat;
  }
  int main()
    int broj;
61
    /* Ucitava se broj sa ulaza */
    scanf("%x", &broj);
    /* Ispisuju se, redom, najveci i najmanji broj formirani od bitova
67
       unetog broja */
```

```
printf("Najveci:\n");
stampaj_bitove(najveci(broj));

printf("Najmanji:\n");
stampaj_bitove(najmanji(broj));

return 0;
}
```

```
#include <stdio.h>
#include "stampanje_bitova.h"
 /* Funckija postavlja na nulu n bitova pocev od pozicije p. */
  unsigned postavi_0(unsigned x, unsigned n, unsigned p)
  Formira se maska cija binarna reprezentacija ima n bitova
   postavljenih na O pocev od pozicije p, dok su svi ostali
  postavljeni na 1. Na primer, za n=5 i p=10 formira se maska oblika
   1111 1111 1111 1111 1111 1000 0011 1111
   To se postize na sledeci nacin:
12
   ~0
                             1111 1111 1111 1111 1111 1111 1111 1111
   (~0 << n)
                             1111 1111 1111 1111 1111 1111 1110 0000
14
   \sim (\sim 0 << n)
                             0000 0000 0000 0000 0000 0000 0001 1111
   (\sim(\sim 0 << n) << (p-n+1)) 0000 0000 0000 0000 0000 0111 1100 0000
    ~(~(~0 << n) << (p-n+1)) 1111 1111 1111 1111 1111 1000 0011 1111
18
   unsigned maska = (((0 << n) << (p - n + 1));
20
   return x & maska;
22 }
24 /* Funckija postavlja na jedinicu n bitova pocev od pozicije p. */
  unsigned postavi_1(unsigned x, unsigned n, unsigned p)
26 {
28
     Formira se maska kod koje je samo n bitova pocev od pocev od
     pozicije p jednako 1, a ostali su 0.
30
     Na primer, za n=5 i p=10 formira se maska oblika
      0000 0000 0000 0000 0000 0111 1100 0000
   unsigned maska = \sim(\sim 0 << n) << (p - n + 1);
   return x | maska;
36
  }
38
```

```
/* Funkcija vraca celobrojno polje bitova, desno poravnato, koje
     predstavlja n bitova pocev od pozicije p u binarnoj
     reprezentaciji broja x. */
  unsigned vrati_bitove(unsigned x, unsigned n, unsigned p)
42
44
    Kreira se maska kod koje su poslednjih n bitova 1, a ostali su 0.
46
      Na primer, za n=5
      0000 0000 0000 0000 0000 0000 0001 1111
48
    unsigned maska = \sim(\sim 0 << n);
    /* Najpre se vrednost promenljive x pomera u desno tako da trazeno
       polje bude uz desni kraj. Zatim se maskiraju ostali bitovi, sem
       zeljenih n i funkcija vraca tako dobijenu vrednost */
54
    return maska & (x >> (p - n + 1));
  1
56
  /* Funkcija vraca broj x kome su n bitova pocev od pozicije p
     postavljeni na vrednosti n bitova najmanje tezine binarne
60
     reprezentacije broja y */
  unsigned postavi_1_n_bitova(unsigned x, unsigned n, unsigned p,
                             unsigned y)
    /* Kreira se maska kod kod koje su poslednjih n bitova 1, a ostali
64
      su 0. */
    unsigned poslednjih_n_1 = \sim(\sim 0 << n);
    /* Kao i kod funkcije postavi_0, i ovde se kreira maska koja ima n
68
       bitova postavljenih na O pocevsi od pozicije p, dok su ostali
       bitovi 1. */
    unsigned srednjih_n_0 = ((0 << n) << (p - n + 1));
72
    /* U promenljivu x_postavi_0 se smesta vrednost dobijena kada se u
       binarnoj reprezentaciji vrednosti promenljive x postavi na 0 n
74
       bitova na pozicijama pocev od p */
    unsigned x_postavi_0 = x & srednjih_n_0;
    /* U promenlijvu y_pomeri_srednje se smesta vrednost dobijena od
       binarne reprezentacije vrednosti promenljive y cijih je n
       bitova najnize tezine pomera tako da stoje pocev od pozicije p.
80
       Ostali bitovi su nule. Sa (y & poslednjih_n_1) postave na O svi
82
       bitovi osim najnizih n */
    unsigned y_pomeri_srednje = (y & poslednjih_n_1) << (p - n + 1);
84
    return x_postavi_0 ^ y_pomeri_srednje;
  }
86
  /* Funkcija invertuje bitove u zapisu broja x pocevsi od pozicije p
     njih n */
90 unsigned invertuj (unsigned x, unsigned n, unsigned p)
```

```
/* Formira se maska sa n jedinica pocev od pozicije p. */
     unsigned maska = \sim(\sim 0 << n) << (p - n + 1):
94
     /* Operator ekskluzivno ili invertuje sve bitove gde je
        odgovarajuci bit maske 1. Ostali bitovi ostaju nepromenjeni. */
96
     return maska ^ x;
98
100 int main()
    unsigned x, p, n, y;
    /* Ucitavaju se vrednosti sa standardnog ulaza */
     printf("Unesite neoznacen ceo broj x:\n");
     scanf("%u", &x);
106
     printf("Unesite neoznacen ceo broj n:\n");
     scanf("%u", &n);
108
     printf("Unesite neoznacen ceo broj p:\n");
     scanf("%u", &p);
    printf("Unesite neoznacen ceo broj y:\n");
    scanf("%u", &y);
     /* Ispisuju se binarne reprezentacije broja x i broja koji se
        dobije kada se primeni funkcija postavi_0 za x, n i p */
     printf("x = %10u %36s = ", x, "");
     stampaj_bitove(x);
     printf("postavi_0(%10u,%6u,%6u)%16s = ", x, n, p, "");
118
     stampaj_bitove(postavi_0(x, n, p));
     printf("\n");
120
     /* Ispisuju se binarne reprezentacije broja x i broja koji se
        dobije kada se primeni funkcija postavi_1 za x, n i p */
     printf("x = 10u \frac{36s}{10u} = x, x, \dots);
     stampaj_bitove(x);
     printf("postavi_1(%10u,%6u,%6u)%16s = ", x, n, p, "");
126
     stampaj_bitove(postavi_1(x, n, p));
     printf("\n");
128
     /* Ispisuju se binarne reprezentacije broja x i broja koji se
130
        dobije kada se primeni funkcija vrati_bitove za x, n i p */
     printf("x = 10u 36s = ", x, "");
     stampaj_bitove(x);
     printf("vrati_bitove(%10u,%6u,%6u)%13s = ", x, n, p, "");
134
     stampaj_bitove(vrati_bitove(x, n, p));
     printf("\n");
136
     /* Ispisuju se binarne reprezentacije brojeva x, y i broja koji se
        dobije kada se primeni funkcija postavi_1_n_bitova za x, n, p */
     printf("x = %10u %36s = ", x, "");
140
     stampaj_bitove(x);
     printf("y = %10u %36s = ", y, "");
142
```

```
#include <stdio.h>
  #include "stampanje_bitova.h"
  /* Funkcija ceo broj x rotira u levo za n mesta. */
  unsigned rotiraj_ulevo(int x, unsigned n)
    unsigned bit_najvece_tezine;
    /* Maska koja ima samo bit na poziciji najvece tezine postavljen
       na 1 je neophodna da bi pre pomeranja u levo za 1 bit na
       poziciji najvece tezine bio sacuvan */
12
    unsigned bit_najvece_tezine_maska =
        1 << (sizeof(unsigned) * 8 - 1);
14
    /* n puta se vrsi rotaciju za jedan bit u levo. U svakoj iteraciji
       se odredi bit na poziciji najvece tezine, a potom se pomera
       binarna reprezentacija trenutne vrednosti promenljive x u levo
       za 1. Nakon toga, bit na poziciji najmanje tezine se postavlja
       na vrednost koju je imao bit na poziciji najvece tezine koji je
20
       istisnut pomeranjem */
    for (i = 0; i < n; i++) {
      bit_najvece_tezine = x & bit_najvece_tezine_maska;
      x = x \ll 1 \mid (bit_najvece_tezine ? 1 : 0);
26
    /* Vraca se dobijena vrednost */
    return x;
30
  /* Funkcija neoznacen broj x rotira u desno za n mesta. */
32 unsigned rotiraj_udesno(unsigned x, unsigned n)
```

```
unsigned bit_najmanje_tezine;
    int i:
36
    /* n puta se ponavlja rotacija u desno za jedan bit. U svakoj
       iteraciji se odredjuje bit na poziciji najmanje tezine broja x,
38
       zatim tako odredjeni bit se pomera u levo tako da bit na
       poziciji najmanje tezine dodje do pozicije najvece tezine.
40
       Zatim, nakon pomeranja binarne reprezentacije trenutne vrednosti
       promenljive x za 1 u desno, bit na poziciji najvece tezine se
42
       postaljva na vrednost vec zapamcenog bita koji je bio na
      poziciji
      najmanje tezine. */
44
    for (i = 0; i < n; i++) {
      bit_najmanje_tezine = x & 1;
46
      x = x >> 1 | bit_najmanje_tezine << (sizeof(unsigned) * 8 - 1);</pre>
48
    /* Vraca se dobijena vrednost */
    return x;
52 }
_{54} /* Verzija funkcije koja broj x rotira u desno za n mesta, gde je
    argument funkcije x oznaceni ceo broj */
56 int rotiraj_udesno_oznaceni(int x, unsigned n)
    unsigned bit_najmanje_tezine;
58
    int i;
    /* U svakoj iteraciji se odredjuje bit na poziciji najmanje tezine
       i smesta u promenljivu bit_najmanje_tezine. Kako je x oznacen
       ceo broj, tada se prilikom pomeranja u desno vrsi aritmeticko
       pomeranje i cuva se znak broja. Dakle, razlikuju se dva slucaja
       u zavisnosti od znaka broja x. Nije dovoljno da se ova provera
       izvrsi pre petlje, s obzirom da rotiranjem u desno na poziciju
       nejvece tezine moze doci i 0 i 1, nezavisno od pocetnog znaka
68
       broja smestenog u promenljivu x. */
    for (i = 0; i < n; i++) {
      bit_najmanje_tezine = x & 1;
      if (x < 0)
72
    Siftovanjem u desno broja koji je negativan dobija se 1 kao bit
74
      na poziciji najvece tezine. Na primer ako je x
76
      1010 1011 1100 1101 1110 0001 0010 001b
        (sa b je oznacen ili 1 ili 0 na poziciji najmanje tezine)
      Onda je sadrzaj promenljive bit_najmanje_tezine:
78
      0000 0000 0000 0000 0000 0000 0000 000ь
      Nakon siftovanja sadrzaja promenljive x za 1 u desno
80
      1101 0101 1110 0110 1111 0000 1001 0001
      Kako bi umesto 1 na poziciji najvece tezine u trenutnoj binarnoj
82
      reprezentaciji x bilo postavljeno b nije dovoljno da se pomeri na
```

```
poziciju najvece tezine jer bi se time dobile 0, a u ovom slucaju
      su potrebne jedinice zbog bitovskog & zato se prvo vrsi
      komplementiranje, a zatim pomeranje
86
      ~bit_najmanje_tezine << (sizeof(int)*8 -1)
      88
      gde B oznacava ~b.
      Potom se ponovo vrsi komplementiranje kako bi se b nalazilo na
90
      poziciji najvece tezine i sve jedinice na ostalim pozicijama
      ~(~bit_najmanje_tezine << (sizeof(int)*8 -1))
92
      *************************
94
        x = (x >> 1) & ~(~bit_najmanje_tezine <<</pre>
                         (sizeof(int) * 8 - 1));
96
      else
        x = (x >> 1) \mid bit_najmanje_tezine << (sizeof(int) * 8 - 1);
98
100
     /* Vraca se dobijena vrednost */
    return x;
104
   int main()
  {
106
    unsigned x, n;
108
     /* Ucitavaju se vrednosti sa standardnog ulaza */
    printf("Unesite neoznacen ceo broj x:");
     scanf("%x", &x);
    printf("Unesite neoznacen ceo broj n:");
     scanf("%x", &n);
114
     /* Ispisuje se binarna reprezentacija broja x */
    printf("x\t\t\t= ");
    stampaj_bitove(x);
118
     /* Testiru se napisane funkcije */
    printf("rotiraj_ulevo(%x,%u)\t\t= ", x, n);
120
     stampaj_bitove(rotiraj_ulevo(x, n));
    printf("rotiraj_udesno(%x,%u)\t\t= ", x, n);
    stampaj_bitove(rotiraj_udesno(x, n));
124
    printf("rotiraj_udesno_oznaceni(%x,%u)\t= ", x, n);
     stampaj_bitove(rotiraj_udesno_oznaceni(x, n));
128
    printf("rotiraj_udesno_oznaceni(-%x,%u)\t= ", x, n);
    stampaj_bitove(rotiraj_udesno_oznaceni(-x, n));
130
    return 0;
```

```
#include <stdio.h>
  #include "stampanje_bitova.h"
3
  /* Funkcija vraca vrednost cija je binarna reprezentacija slika u
     ogledalu binarne reprezentacije broja x. */
  unsigned ogledalo(unsigned x)
7
    unsigned najnizi_bit;
9
    unsigned rezultat = 0;
    int i;
    /* U svakoj iteraciji najnizi bit u binarnoj reprezentaciji tekuce
       vrednosti broja x se odredjuje i pamti u promenljivoj
13
       najnizi_bit, nakon cega se na promenljivu x primeni pomeranje u
       desno */
    for (i = 0; i < sizeof(x) * 8; i++) {
      najnizi_bit = x & 1;
      x >>= 1;
      /* Potiskivanjem trenutnog rezultata ka levom kraju svi
19
         prethodno postavljeni bitovi dobijaju vecu poziciju. Novi bit
         se postavlja na najnizu poziciju */
      rezultat <<= 1;
      rezultat |= najnizi_bit;
    /* Vraca se dobijena vrednost */
    return rezultat;
29
  int main()
  {
    int broj;
    /* Ucitava se broj sa ulaza */
    scanf("%x", &broj);
35
    /* Ispisuje se njegova binarna reprezentacija */
    stampaj_bitove(broj);
    /* Ispisuje se i binarna reprezentacija broja dobijenog pozivom
       funkcije ogledalo */
41
    stampaj_bitove(ogledalo(broj));
43
    return 0:
  }
45
```

```
#include <stdio.h>
  /* Funkcija vraca 1 ukoliko je u binarnoj reprezentaciji broja n
     broj jedinica veci od broja nula. U suprotnom funkcija vraca 0 */
  int broj_01(unsigned int n)
6
    int broj_nula, broj_jedinica;
    unsigned int maska;
    broj_nula = 0;
    broj_jedinica = 0;
12
    /* Maska je inicijalizovana tako da moze da analizira bit najvece
       tezine */
14
    maska = 1 << (sizeof(unsigned int) * 8 - 1);</pre>
    /* Cilj je proci kroz sve bitove broja x, zato se maska u svakoj
       iteraciji pomera u desno pa ce jedini bit koji je postavljen na
18
       1 biti na svim pozicijama u binarnoj reprezentaciji maske */
    while (maska != 0) {
20
      /* Provera da li se na poziciji koju odredjuje maska nalazi 0
         ili 1 i uveca se odgovarajuci brojac */
      if (n & maska) {
        broj_jedinica++;
24
      } else {
        broj_nula++;
26
28
      /* Pomera se maska u desnu stranu */
      maska = maska >> 1;
30
    /* Ako je broj jedinica veci od broja nula funkcija vraca 1, u
       suprotnom vraca 0 */
34
    return (broj_jedinica > broj_nula) ? 1 : 0;
36
  int main()
38
    unsigned int n;
40
    /* Ucitava se broj */
42
    scanf("%u", &n);
44
    /* Ispisuje se rezultat */
    printf("%d\n", broj_01(n));
46
    return 0;
48
  }
```

```
#include <stdio.h>
  /* Funkcija broji koliko se puta dve uzastopne jedinice pojavljuju u
     binarnom zapisu celog čneoznaenog broja x */
4
  int broj_parova(unsigned int x)
  {
6
    int broj_parova;
    unsigned int maska;
8
    /* Vrednost promenljive koja predstavlja broj parova se
       inicijalizuje na 0 */
    broj_parova = 0;
12
    /* Postavlja se maska tako da moze da procita da li su dva
14
       najmanja bita u zapisu broja x 11 */
    /* Binarna reprezentacija broja 3 je 000....00011 */
    maska = 3;
18
    while (x != 0) {
      /* Provera da li se na najmanjim pozicijama broj x nalazi 11 par
20
      if ((x & maska) == maska) {
        broj_parova++;
24
      /* Pomera se broj u desnu stranu da bi se u narednoj iteraciji
26
         proveravao sledeci par bitova. Pomeranjem u desno bit najvece
         tezine se popunjava nulom jer je x neoznacen broj */
28
      x = x \gg 1;
30
    /* Vraca se dobijena vrednost */
    return broj_parova;
  }
34
  int main()
36
   unsigned int x;
38
    /* Ucitava se broj */
40
    scanf("%u", &x);
42
    /* Ispisuje se rezultat */
    printf("%d\n", broj_parova(x));
44
    return 0;
46
```

```
#include <stdio.h>
  /* Niska koja se formira je duzine (sizeof(unsigned int)*8)/4 +1 jer
     su za svaku heksadekadnu cifru potrebne 4 binarne cifre i jedna
     dodatna pozicija za terminirajucu nulu. Prethodni izraz se moze
     zapisati kao sizeof(unsigned int)*2+1. */
  #define MAKS_DUZINA sizeof(unsigned int)*2 +1
  /* Funkcija za neoznacen broj x formira nisku s koja sadrzi njegov
    heksadekadni zapis */
  void prevod(unsigned int x, char s[])
12 {
    int i;
14
    unsigned int maska;
    int vrednost;
    /* Heksadekadni zapis broja 15 je 000...0001111 - odgovarajuca
       maska za citanje 4 uzastopne cifre */
    maska = 15;
20
      Broj se posmatra od pozicije najmanje tezine ka poziciji najvece
      tezine. Na primer za broj cija je binarna reprezentacija
      0000000001101000100001111010101
      u prvom koraku se citaju bitovi izdvojeni sa <...>:
      000000000110100010000111101<0101>
      u drugom koraku:
      00000000011010001000011<1101>0101
      u trecem koraku:
      0000000001101000100<0011>11010101 i tako redom...
      Indeks i oznacava poziciju na koju se smesta vrednost.
    for (i = MAKS_DUZINA - 2; i \ge 0; i--) {
      /* Vrednost izdvojene cifre */
      vrednost = x & maska;
      /* Ako je vrednost iz opsega od 0 do 9 odgovarajuci karakter se
38
         dobija dodavanjem ASCII koda 'O'. Ako je vrednost iz opsega
         od 10 do 15 odgovarajuci karakter se dobija tako sto se prvo
         oduzme 10 (time se dobiju vrednosti od 0 do 5) pa se na tako
         dobijenu vrednost doda ASCII kod 'A' (time se dobija
42
         odgovarajuce slovo 'A', 'B', ... 'F') */
      if (vrednost < 10) {
        s[i] = vrednost + '0';
      } else {
46
        s[i] = vrednost - 10 + 'A';
48
      /* Primenljiva x se pomera za 4 bita u desnu stranu i time se u
```

```
narednoj iteraciji posmatraju sledeca 4 bita */
      x = x \gg 4;
54
    /* Upisuje se terminirajuca nula */
    s[MAKS_DUZINA - 1] = '\0';
56
58
  int main()
60 {
    unsigned int x;
   char s[MAKS_DUZINA];
62
    /* Ucitava se broj sa ulaza */
64
    scanf("%u", &x);
66
    /* Poziva se funkcija za prevodjenje */
    prevod(x, s);
68
    /* I stampa se dobijena niska */
    printf("%s\n", s);
    return 0;
74 }
```

```
#include <stdio.h>
 #include <stdlib.h>
   Resenje linearne slozenosti:
   x^0 = 1
   x^k = x * x^(k-1)
                int stepen(int x, int k)
10 {
   if (k == 0)
    return 1;
14
  return x * stepen(x, k - 1);
   /* kraci zapis: return k == 0 ? 1 : x * stepen(x,k-1); */
16 }
  18
    Resenje logaritamske slozenosti:
    x^0 = 1;
20
    x^k = x * (x^2)^(k/2), za neparno k
    x^k = (x^2)^(k/2), za parno k
    Ovom resenju ce biti potrebno manje rekurzivnih poziva da bi
24
    se doslo do rezultata, i stoga je efikasnije.
```

```
int stepen_2(int x, int k)
26
    if (k == 0)
28
      return 1;
30
    /* Ako je stepen paran */
    if ((k \% 2) == 0)
     return stepen_2(x * x, k / 2);
34
    /* Inace (ukoliko je stepen neparan) */
   return x * stepen_2(x * x, k / 2);
36
38
  int main()
  {
40
    int x, k, ind;
42
    /* Ucitavanje rednog broja funkcije koja ce se primeniti */
    printf("Unesite redni broj funkcije (1/2):\n");
44
    scanf("%d", &ind);
46
    /* Ucitavanje vrednosti za x i k */
    printf("Unesite broj x:\n");
48
    scanf("%d", &x);
    printf("Unesite broj k:\n");
    scanf("%d", &k);
    /* Ispisuje se vrednost koju vraca odgovarajuca funkcija */
    if (x == 1)
54
      printf("%d\n", stepen(x, k));
    else if (x == 2)
56
      printf("%d\n", stepen_2(x, k));
    else {
58
      fprintf(stderr, "Greska: Neodgovarajuci redni broj funkcije.\n");
      exit(EXIT_FAILURE);
60
62
    exit(EXIT_SUCCESS);
  }
```

```
#include <stdio.h>

/* NAPOMENA: Ovaj problem je iskoriscen da ilustruje uzajamnu
(posrednu) rekurziju */

/* Deklaracija funkcije neparan mora da bude navedena jer se ta
funkcija koristi u telu funkcije paran, tj. koristi se pre svoje
definicije. Funkcija je mogla biti deklarisana i u telu funkcije
```

```
paran. */
10 unsigned neparan(unsigned n);
12 /* Funkcija paran vraca 1 ako broj n ima paran broj cifara, inace
     vraca 0 */
14 unsigned paran(unsigned n)
    if (n <= 9)
16
      return 0;
    else
18
      return neparan(n / 10);
20 }
22 /* Funkcija neparan vraca 1 ako broj n ima neparan broj cifara,
     inace vraca 0 */
24 unsigned neparan(unsigned n)
    if (n <= 9)
26
      return 1;
    else
28
      return paran(n / 10);
30 }
32 int main()
    int n;
34
    /* Ucitava se ceo broj */
36
    scanf("%d", &n);
38
    /* Ispisuje se rezultat */
    printf("Uneti broj ima %sparan broj cifara.\n",
40
            (paran(n) == 1 ? "" : "ne"));
42
    return 0;
44 }
```

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

/* Pomocna funkcija koja izracunava n! * result. Koristi repnu
rekurziju. Result je argument u kome se akumulira do tada
izracunatu vrednost faktorijela. Kada dodje do izlaza iz
rekurzije iz rekurzije potrebno je da vratimo result. */
int faktorijel_repna(int n, int result)
{
   if (n == 0)
        return result;
}
```

```
13
    return faktorijel_repna(n - 1, n * result);
  /* U sledecim funkcijama je prikazan postupak oslobadjanja od repne
     rekurzije koja postoji u funkciji faktorijel_repna. */
17
  /* Funkcija se transformise tako sto se rekurzivni poziv zemeni sa
19
     naredbama kojima se vrednost argumenta funkcije postavlja na
     vrednost koja bi se prosledjivala rekurzivnom pozivu i
     navodjenjem goto naredbe za vracanje na pocetak tela funkcije. */
23 int faktorijel_repna_v1(int n, int result)
  pocetak:
25
    if (n == 0)
      return result;
    result = n * result;
29
    n = n - 1:
    goto pocetak;
  /* Pisanje bezuslovnih skokova (goto naredbi) nije dobra
     programerska praksa i prethodna funkcija se koristi samo kao
     medjukorak. Sledi iterativno resenje bez bezuslovnih skokova */
  int faktorijel_repna_v2(int n, int result)
37
    while (n != 0) {
39
      result = n * result;
      n = n - 1;
41
43
    return result;
  }
45
  /* Prilikom poziva prethodnih funkcija pored prvog argumenta celog
     broja n, mora da se salje i 1 za vrednost drugog argumenta u kome
     ce se akumulirati rezultat. Funkcija faktorijel(n) je ovde radi
49
     udobnosti korisnika, jer je sasvim prirodno da za faktorijel
     zahteva samo 1 parametar. Funkcija faktorijel izracunava n!, tako
     sto odgovarajucoj gore navedenoj funkciji koja zaista racuna
     faktorijel, salje ispravne argumente i vraca rezultat koju joj ta
     funkcija vrati. Za testiranje, zameniti u telu funkcije faktorijel
     poziv faktorijel_repna sa pozivom faktorijel_repna_v1, a zatim sa
     pozivom funkcije faktorijel_repna_v2. */
  int faktorijel(int n)
    return faktorijel_repna(n, 1);
59
  int main()
  {
    int n;
```

```
/* Ucitava se ceo broj */
printf("Unesite n (<= 12): ");
scanf("%d", &n);
if (n > 12) {
    fprintf(stderr, "Greska: Nedozvoljena vrednost za n.\n");
    exit(EXIT_FAILURE);
}

/* Ispisuje se rezultat */
printf("%d! = %d\n", n, faktorijel(n));

exit(EXIT_SUCCESS);
}
```

```
1 #include <stdio.h>
3 /* a) Funkcija racuna n-ti element u nizu F - iterativna verzija */
  int f_iterativna(int n, int a, int b)
5 {
    int i;
  int f_0 = 0;
   int f_1 = 1;
9
   int tmp;
   if (n == 0)
     return 0;
   for (i = 2; i <= n; i++) {
     tmp = a * f_1 + b * f_0;
15
     f_0 = f_1;
      f_1 = tmp;
17
19
    return f_1;
21 }
23 /* b) Funkcija racuna n-ti element u nizu F - rekurzivna verzija */
  int f_rekurzivna(int n, int a, int b)
25 {
    /* Izlaz iz rekurzije */
27
  if (n == 0 || n == 1)
      return n;
29
   /* Rekurzivni pozivi */
  return a * f_rekurzivna(n - 1, a, b) +
        b * f_rekurzivna(n - 2, a, b);
33 }
```

```
35 /* NAPOMENA: U slucaju da se rekurzijom problem svodi na vise manjih
     podproblema koji se mogu preklapati, postoji opasnost da se
     pojedini podproblemi manjih dimenzija resavaju veci broj puta.
     Npr. F(20) = a*F(19) + b*F(18), a F(19) = a*F(18) + b*F(17), tj.
     problem F(18) se resava dva puta! Problemi manjih dimenzija ce se
39
     resavati jos veci broj puta. Resenje za ovaj problem je
     kombinacija rekurzije sa dinamickim programiranjem. Podproblemi se
41
     resavaju samo jednom, a njihova resenja se pamte u memoriji
     (obicno u nizovima ili matricama), odakle se koriste ako tokom
     resavanja ponovo budu potrebni.
45
     U narednoj funkciji vec izracunati clanovi niza se cuvaju u
     statickom nizu celih brojeva, jer se staticki niz ne smesta na
47
     stek, kao sto je to slucaj sa lokalnim promenljivama, vec na
     segment podataka, odakle je dostupan svim pozivima rekurzivne
49
     funkcije. */
  /* c) Funkcija racuna n-ti broj niza F - efikasnija rekurzivna
     verzija */
  int f_napredna(int n, int a, int b)
    /* Niz koji cuva resenja podproblema. Kompajler inicijalizuje
       staticke promenljive na podrazumevane vrednosti. Stoga,
       elemente celobrojnog niza inicijalizuje na 0 */
    static int f[20];
59
    /* Ako je podproblem vec ranije resen, koristi se resenje koje je
       vec izracunato */
    if (f[n] != 0)
      return f[n];
    /* Izlaz iz rekurzije */
    if (n == 0 || n == 1)
      return f[n] = n;
    /* Rekurzivni pozivi */
    return f[n] =
        a * f_napredna(n - 1, a, b) + b * f_napredna(n - 2, a, b);
73
  int main()
    int n, a, b, ind;
    /* Unosi se redni broj funkcije koja ce se primeniti */
    printf("Unesite redni broj funkcije:\n");
    printf("1 - iterativna\n");
81
    printf("2 - rekurzivna\n");
    printf("3 - rekurzivna napredna\n");
83
    scanf("%d", &ind);
85
    /* Ucitaju se koeficijenti a i b */
```

```
printf("Unesite koeficijente:\n");
    scanf("%d%d", &a, &b);
89
    /* Ucitava se broj n */
    printf("Unesite koji clan niza se racuna:\n");
91
    scanf("%d", &n);
93
    /* Na osnovu vrednosti promenljive ind ispisuje se rezultat poziva
       funkcije f_iterativna, f_rekurzivna ili f_napredna */
95
    if (ind == 0)
      printf("F(%d) = %d\n", n, f_iterativna(n, a, b));
97
    else if (ind == 1)
      printf("F(%d) = %d\n", n, f_rekurzivna(n, a, b));
99
    else
      printf("F(%d) = %d n", n, f_napredna(n, a, b));
    return 0;
```

```
#include <stdio.h>
  /* Funkcija odredjuje zbir cifara zadatog broja x */
  int zbir_cifara(unsigned int x)
    /* Izlazak iz rekurzije: ako je broj jednocifren */
6
    if (x < 10)
     return x;
    /* Zbir cifara broja jednak je zbiru svih njegovih cifara osim
       poslednje cifre + poslednja cifra tog broja */
    return zbir_cifara(x / 10) + x % 10;
12
14
  int main()
16 {
    unsigned int x;
18
    /* Ucitava se ceo broj */
   scanf("%u", &x);
20
    /* Ispisuje se zbir cifara ucitanog broja */
    printf("%d\n", zbir_cifara(x));
24
    return 0;
26 }
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAKS_DIM 100
  /* Ako je n<=0, onda je suma niza jednaka nuli. Ako je n>0, onda je
     suma niza jednaka sumi prvih n-1 elementa uvecenoj za poslednji
     element niza. */
  int suma_niza_1(int *a, int n)
    if (n \le 0)
      return 0;
    return suma_niza_1(a, n - 1) + a[n - 1];
14 }
  /* Funkcija napisana na drugi nacin: Ako je n<=0, onda je suma niza
     jednaka nuli. Ako je n>0, suma niza je jednaka zbiru prvog
     elementa niza i sume preostalih n-1 elementa. */
  int suma_niza_2(int *a, int n)
    if (n <= 0)
      return 0;
22
    return a[0] + suma_niza_2(a + 1, n - 1);
  }
  int main()
    int a[MAKS_DIM];
30
    int n, i = 0, ind;
    /* Ucitava se redni broj funkcije */
    printf("Unesite redni broj funkcije (1 ili 2):\n");
    scanf("%d", &ind);
    /* Ucitava se broj elemenata niza */
    printf("Unesite dimenziju niza:\n");
    scanf("%d", &n);
38
    /* Ucitava se n elemenata niza. */
    printf("Unesite elemente niza:\n");
    for (i = 0; i < n; i++)
42
      scanf("%d", &a[i]);
    /* Na osnovu vrednosti promenljive ind ispisuje se rezultat poziva
       funkcije suma_niza_1, ondosno suma_niza_2 */
46
    if (ind == 1)
      printf("Suma elemenata je %d\n", suma_niza_1(a, n));
48
    else if (ind == 2)
      printf("Suma elemenata je %d\n", suma_niza_2(a, n));
```

```
else {
    fprintf(stderr, "Greska: Neodgovarajuci redni broj funkcije.\n");
    exit(EXIT_FAILURE);
}

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
2 #define MAKS_DIM 256
  /* Rekurzivna funkcija koja odredjuje maksimum celobrojnog niza niz
     dimenzije n */
6 int maksimum_niza(int niz[], int n)
    /* Izlazak iz rekurzije: ako je niz dimenzije jedan, najveci je
      ujedno i jedini element niza */
    if (n == 1)
      return niz[0];
    /* Resava se problem manje dimenzije */
    int max = maksimum_niza(niz, n - 1);
16
    /* Na osnovu poznatog resenja problema dimenzije n-1, resava se
       problem dimenzije n */
    return niz[n-1] > max ? niz[n-1] : max;
18
  }
20
  int main()
    int brojevi[MAKS_DIM];
   int n;
26
    /* Sve dok se ne stigne do kraja ulaza, brojevi se ucitavaju u
       niz. Promenljiva i predstavlja indeks tekuceg broja. U niz se
28
       ne moze ucitati vise od MAKS_DIM brojeva, pa se u slucaju da
       promenljiva i dostigne vrednost MAKS_DIM prekida unos novih
30
       brojeva. */
    int i = 0;
    while (scanf("%d", &brojevi[i]) != EOF) {
      if (i == MAKS_DIM)
34
        break;
    }
36
    n = i;
38
    /* Stampa se maksimum unetog niza brojeva */
    printf("%d\n", maksimum_niza(brojevi, n));
40
```

```
42 return 0; }
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #define MAKS_DIM 256
  /* Funkcija koja izracunava skalarni proizvod dva data vektora */
  int skalarno(int a[], int b[], int n)
    /* Izlazak iz rekurzije: vektori su duzine 0 */
    if (n == 0)
      return 0;
    /* Na osnovu resenja problema dimenzije n-1, resava se problem
       dimenzije n primenom definicije skalarnog proizvoda a*b =
13
       a[0]*b[0] + a[1]*b[1] + ... + a[n-2]*a[n-2] + a[n-1]*a[n-1]
       Dakle, skalarni proizvod dva vektora duzine n se dobija kada se
       na skalarni proizvod dva vektora duzine n-1 koji se dobiju od
       polazna dva vektora otklanjanjem poslednjih elemenata, doda
       proizvod poslednja dva elementa polaznih vektora. */
    else
      return skalarno(a, b, n - 1) + a[n - 1] * b[n - 1];
  int main()
    int i, a[MAKS_DIM], b[MAKS_DIM], n;
    /* Unosi se dimenzija nizova */
    printf("Unesite dimenziju nizova:");
    scanf("%d", &n);
29
    /* Provera da li je dimenzija niza odgovarajuca */
    if (n < 0 \mid \mid n > MAKS_DIM) {
33
      fprintf(stderr,
               "Greska: Dimenzija mora biti prirodan broj <= %d!\n",
35
              MAKS_DIM);
      exit(EXIT_FAILURE);
37
    /* A zatim i elementi nizova */
    printf("Unesite elemente prvog niza:");
    for (i = 0; i < n; i++)
41
      scanf("%d", &a[i]);
43
    printf("Unesite elemente drugog niza:");
    for (i = 0; i < n; i++)
45
      scanf("%d", &b[i]);
```

```
/* Ispisuje se rezultat skalarnog proizvoda dva ucitana niza */
printf("Skalarni proizvod je %d\n", skalarno(a, b, n));

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
2 #define MAKS_DIM 256
 /* Funkcija koja racuna broj pojavljivanja elementa x u nizu a
     duzine n */
 int br_pojave(int x, int a[], int n)
    /* Izlazak iz rekurzije: za niz duzine jedan broj pojava broja x u
      nizu je 1 ukoliko je jedini element a[0] bas x ili 0 inace */
    if (n == 1)
     return a[0] == x ? 1 : 0;
12
    /* U promenljivu bp se smesta broj pojave broja x u prvih n-1
       elemenata niza a. Ukupan broj pojavljivanja broja x u celom
14
       nizu a je jednak bp uvecanom za jedan ukoliko je se na poziciji
       n-1 u nizu a nalazi broj x */
16
    int bp = br_pojave(x, a, n - 1);
    return a[n - 1] == x ? 1 + bp : bp;
18
  }
20
  int main()
22 1
    int x, a[MAKS_DIM];
   int n, i = 0;
    /* Ucitava se ceo broj */
    printf("Unesite ceo broj:");
    scanf("%d", &x);
    /* Sve dok se ne stigne do kraja ulaza, ucitavaju se brojevi u
30
       niz. Promenljiva i predstavlja indeks tekuceg broja. U niz se
       ne moze ucitati vise od MAKS_DIM brojeva, pa se u slucaju da
       promenljiva i dostigne vrednost MAKS_DIM prekida unos novih
34
       brojeva. */
    printf("Unesite elemente niza:");
    i = 0;
36
    while (scanf("%d", &a[i]) != EOF) {
38
      if (i == MAKS_DIM)
        break;
40
42
    n = i;
```

```
/* Ispisuje se broj pojavljivanja */
printf("Broj pojavljivanja je %d\n", br_pojave(x, a, n));

return 0;
}
```

```
1 #include <stdio.h>
  #define MAKS_DIM 256
  /* Funkcija koja proverava da li su tri zadata broja uzastopni
     clanovi niza */
  int tri_uzastopna_clana(int x, int y, int z, int a[], int n)
    /* Ako niz ima manje od tri elementa izlazi se iz rekurzije i
       vraca se 0 jer nije ispunjeno da su x, y i z uzastopni clanovi
       niza */
    if (n < 3)
      return 0;
    /* Da bi bilo ispunjeno da su x, y i z uzastopni clanovi niza a
       dovoljno je da su oni poslednja tri clana niza ili da se oni
       rekuzivno tri uzastopna clana niza a bez poslednjeg elementa */
    return ((a[n-3] == x) && (a[n-2] == y) && (a[n-1] == z))
        || tri_uzastopna_clana(x, y, z, a, n - 1);
  }
  int main()
    int x, y, z, a[MAKS_DIM];
    /* Ucitavaju se tri cela broja za koje se ispituje da li su
27
       uzastopni clanovi niza */
    printf("Unesite tri cela broja:");
29
    scanf("%d%d%d", &x, &y, &z);
    /* Sve dok se ne stigne do kraja ulaza, brojeve se ucitavaju u
       niz. Promenljiva i predstavlja indeks tekuceg broja. U niz se
       ne moze ucitati vise od MAKS_DIM brojeva, pa se u slucaju da
33
       promenljiva i dostigne vrednost MAKS_DIM prekida unos novih
       brojeva. */
35
    printf("Unesite elemente niza:");
    int i = 0;
    while (scanf("%d", &a[i]) != EOF) {
      if (i == MAKS_DIM)
        break;
41
```

```
#include <stdio.h>
  /* Funkcija koja broji bitove postavljene na 1. */
4 int prebroj(int x)
    /* Izlaz iz rekurzije */
    if (x == 0)
      return 0;
    /* Ukoliko vrednost promenljive x nije 0, neki od bitova broja x
       je postavljen na 1. Koriscenjem odgovarajuce maske proverava se
12
       vrednost bita na poziciji najvece tezine i na osnovu toga se
       razlikuju dva slucaja. Ukoliko je na toj poziciji nula, onda je
       broj jedinica u zapisu x isti kao broj jedinica u zapisu broja
14
       x<<1, jer se pomeranjem u levo sa desne stane dopisuju 0. Ako je
       na poziciji najvece tezine jedinica, rezultat dobijen
16
       pozivom funkcije za x<<1 treba uvecati za jedan. Za rekurzivni
       poziv se salje vrednost koja se dobija kada se x pomeri u levo.
18
       Napomena: argument funkcije x je oznacen ceo broj, usled cega se
       ne koristi pomeranje udesno, jer funkciji moze biti prosledjen i
20
       negativan broj. Iz tog razloga, odlucujemo se da proveramo
       najvisi, umesto najnizeg bita */
    if (x & (1 << (sizeof(x) * 8 - 1)))
      return 1 + prebroj(x << 1);
24
26
      return prebroj(x << 1);
     Krace zapisano
28
      return ((x\& (1<<(sizeof(x)*8-1))) ? 1 : 0) + prebroj(x<<1);
30
  int main()
34 {
    int x;
36
```

```
/* Ucitava se ceo broj */
scanf("%x", &x);

/* Ispisuje se rezultat */
printf("%d\n", prebroj(x));

return 0;

44
}
```

```
#include <stdio.h>
  /* Rekurzivna funkcija za odredjivanje najvece oktalne cifre */
  int maks_oktalna_cifra(unsigned x)
    /* Izlazak iz rekurzije: ako je vrednost broja 0, onda je i
       vrednost najvece oktalne cifre u broju 0 */
    if (x == 0)
      return 0;
    /* Odredjuje se poslednja oktalna cifra u broju */
    int poslednja_cifra = x & 7;
    /* Odredjuje se maksimalna oktalna cifra u broju kada se iz njega
14
       izbrise poslednja oktalna cifra */
    int maks_bez_poslednje_cifre = maks_oktalna_cifra(x >> 3);
    return poslednja_cifra > maks_bez_poslednje_cifre ? poslednja_cifra
18
        : maks_bez_poslednje_cifre;
20
  int main()
    unsigned x;
    /* Ucitava se neoznacen ceo broj */
26
    scanf("%u", &x);
28
    /* Ispisuje se vrednost najvece oktalne cifre unetog broja */
    printf("%d\n", maks_oktalna_cifra(x));
30
    return 0;
```

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

```
/* Rekurzivna funkcija za odredjivanje najvece heksadekadne cifre */
4 int maks_heksadekadna_cifra(unsigned x)
    /* Izlazak iz rekurzije: ako je vrednost broja 0, onda je i
6
       vrednost najvece heksadekadne cifre u broju 0 */
    if (x == 0)
     return 0;
    /* Odredjuje se poslednja heksadekadna cifra u broju */
    int poslednja_cifra = x & 15;
12
    /* Odredjuje se maksimalna heksadekadna cifra broja kada se iz
14
       njega izbrise poslednja heksadekadna cifra */
    int maks_bez_poslednje_cifre = maks_heksadekadna_cifra(x >> 4);
    return poslednja_cifra > maks_bez_poslednje_cifre ? poslednja_cifra
18
        : maks_bez_poslednje_cifre;
20
  }
22 int main()
    unsigned x;
24
    /* Ucitava se neoznacen ceo broj */
26
    scanf("%u", &x);
28
    /* Ispisivanje vrednosti najvece heksadekadne cifre unetog broja */
    printf("%d\n", maks_heksadekadna_cifra(x));
30
    return 0;
```

```
#include <stdio.h>
2 #include <string.h>
4 /* Niska moze imati najvise 31 karaktera + 1 za terminalnu nulu */
  #define MAKS_DIM 32
  /* Funkcija ispituje da li je zadata niska duzine n palindrom */
8 int palindrom(char s[], int n)
10
    /* Izlaz iz rekurzije - trivijalno, niska duzine 0 ili 1 je
       palindrom */
    if ((n == 1) || (n == 0))
12
      return 1;
14
    /* Da bi niska bila palindrom potrebno je da se poklapaju prvi i
       poslednji karakter i da je palindrom niska koja nastaje kada se
       polaznoj nisci otklone prvi i poslednji karakter */
```

```
return (s[n-1] == s[0]) \&\& palindrom(s+1, n-2);
20
  int main()
22 | {
    char s[MAKS_DIM];
    int n;
24
    /* Ucitava se niska sa standardnog ulaza */
26
    scanf("%s", s);
28
    /* Odredjuje se duzina niske */
    n = strlen(s);
30
    /* Ispisuje se da li je niska palindrom ili nije */
32
    if (palindrom(s, n))
      printf("da\n");
34
    else
      printf("ne\n");
36
    return 0;
38
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAKS_DUZINA_PERMUTACIJE 15
  /* Funkcija koja ispisuje elemente niza a duzine n */
  void ispisi_niz(int a[], int n)
  {
    int i;
    for (i = 1; i <= n; i++)
      printf("%d ", a[i]);
    printf("\n");
14
  /* Funkcija koja vraca 1 ako se broj x nalazi u nizu a duzine n, a
     inace 0 */
  int koriscen(int a[], int n, int x)
18
    int i;
20
    /* Obilaze se svi elementi niza */
    for (i = 1; i <= n; i++)
22
      /* Ukoliko se naidje na trazenu vrednost, pretraga se prekida */
      if (a[i] == x)
24
        return 1;
26
```

```
/* Zakljucuje se da broj nije pronadjen */
    return 0;
30
  /* Funkcija koja ispisuje sve permutacije od skupa {1,2,...,n}
     dobija kao argument niz a[] u koji se smesta permutacija, broj m
     oznacava da se u okviru tog poziva funkcije na m-tu poziciju u
     permutaciji smesta jedan od preostalih celih brojeva, n je
34
     velicina skupa koji se permutuje. Funkciju se inicijalno poziva
     sa argumentom m = 1, jer formiranje permutacije pocinje od
36
     pozicije broj 1. Stoga, a[0] se ne koristi. */
  void permutacija(int a[], int m, int n)
38
    int i;
40
    /* Izlaz iz rekurzije: Ako je pozicija na koju treba smestiti broj
42
       premasila velicinu skupa, onda se svi brojevi vec nalaze u
       permutaciji i ispisuje se permutacija. */
44
    if (m > n) {
      ispisi_niz(a, n);
46
      return;
48
    /* Ideja: pronalazi se prvi broj koji moze da se postavi na m-to
       mesto u nizu (broj koji se do sada nije pojavio u permutaciji).
       Zatim, rekurzivno se pronalaze one permutacije koje odgovaraju
       ovako postavljenom pocetku permutacije. Kada se to zavrsi, vrsi
       se provera da li postoji jos neki broj koji moze da se stavi na
       m-to mesto u nizu (to se radi u petlji). Ako ne postoji,
       funkcija zavrsava sa radom. Ukoliko takav broj postoji, onda se
56
       ponovo poziva rekurzivno pronalazenje odgovarajucih permutacija,
       ali sada sa drugacije postavljenim prefiksom. */
58
    for (i = 1; i <= n; i++) {
      /* Ako se broj i nije do sada pojavio u permutaciji od 1 do m-1
         pozicije, onda se on postavlja na poziciju m i poziva se
         ponovo funkcija da dopuni ostatak permutacije posle
         upisivanja i na poziciju m. Inace, nastavlja se dalje,
         trazeci broj koji se nije pojavio do sada u permutaciji. */
      if (!koriscen(a, m - 1, i)) {
        a[m] = i:
        permutacija(a, m + 1, n);
68
  }
70
72 int main(void)
74
    int n;
    int a[MAKS_DUZINA_PERMUTACIJE + 1];
76
    /* Ucitava se broj n iz odgovarajuceg opsega */
    scanf("%d", &n);
```

#### Rešenje 1.34

```
#include <stdio.h>
  #include <stdlib.h>
4 /* Rekurzivna funkcija za racunanje binomnog koeficijenta */
  int binomni_koeficijent(int n, int k)
    /* Ukoliko je k=0 ili k=n, onda je binomni koeficijent 0. Ukoliko
       je k strogo izmedju 0 i n, onda se koristi formula bk(n,k) =
       bk(n-1,k-1) + bk(n-1,k) koja se moze izvesti iz definicije
       binomnog koeficijenata */
    return (0 < k && k < n) ?
       binomni_koeficijent(n - 1, k - 1) +
        binomni_koeficijent(n - 1, k) : 1;
14 }
  Iterativno izracunavanje datog binomnog koeficijenta
    int binomni_koeficijent (int n, int k) {
20
     int i, j, b;
      for (b=i=1, j=n; i<=k; b =b * j-- / i++);
     return b;
26
    Iterativno resenje je efikasnije i preporucuje se. Rekurzivno
    resenje je navedeno u cilju demonstracije rekurzivnih tehnika.
30
  /* Svaki element n-te hipotenuze (osim ivicnih jedinica) dobija kao
     zbir 2 elementa iz n-1 hipotenuze. Ukljucujuci i pomenute dve
     ivicne jedinice suma elemenata n-te hipotenuze je tacno 2 puta
    veca od sume elemenata prethodne hipotenuze. */
  int suma_elemenata_hipotenuze(int n)
```

```
36 {
    return n > 0 ? 2 * suma_elemenata_hipotenuze(n - 1) : 1;
40 int main()
  {
    int n, k, i, d, r;
42
    /* Ucitavaju se brojevi d i r */
44
    scanf("%d %d", &d, &r);
46
    /* Ispisuje se Paskalov trougao */
    putchar('\n');
48
    for (n = 0; n <= d; n++) {
     for (i = 0; i < d - n; i++)
       printf(" ");
     for (k = 0; k <= n; k++)
        printf("%4d", binomni_koeficijent(n, k));
      putchar('\n');
54
56
    /* Proverava se da li je r nenegativan */
    if (r < 0) {
     fprintf(stderr,
              "Redni broj hipotenuze mora biti veci ili jednak od 0!\n"
      exit(EXIT_FAILURE);
    /* Ispisuje se suma elemenata hipotenuze */
64
    printf("%d\n", suma_elemenata_hipotenuze(r));
66
    exit(EXIT_SUCCESS);
68 }
```

# Pokazivači

## 2.1 Pokazivačka aritmetika

**Zadatak 2.1** Za dati celobrojni niz dimenzije n, napisati funkciju koja obrće njegove elemente:

- (a) korišćenjem indeksne sintakse,
- (b) korišćenjem pokazivačke sintakse.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenziju niza n (0 <  $n \le 100$ ), a zatim elemente niza. Pozvati funkciju koja obrće njegove elemente korišćenjem indeksne sintakse i prikazati sadržaj niza. Nakon toga pozvati funkciju koja obrće njegove elemente korišćenjem pokazivačke sintakse i prikazati sadržaj niza.

```
Primer 1

| Interakcija sa programom: | Interakcija sa programom: | Unesite dimenziju niza: 3 | Unesite dimenziju niza: 0 | Izlaz za greške: 1 -2 3 | Nakon obrtanja elemenata, niz je: 3 -2 1 | Nakon ponovnog obrtanja elemenata, niz je: 3 -2 1
```

**Z**adatak 2.2 Dat je niz realnih brojeva dimenzije n. Korišćenjem pokazivačke sintakse, napisati:

(a) funkciju zbir koja izračunava zbir elemenata niza,

- (b) funkciju proizvod koja izračunava proizvod elemenata niza,
- (c) funkciju min\_element koja izračunava najmanji elemenat niza,
- (d) funkciju max\_element koja izračunava najveći elemenat niza.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenziju n ( $0 < n \le 100$ ) realnog niza, a zatim i elemente niza. Na standardni izlaz ispisati zbir, proizvod, minimalni i maksimalni element učitanog niza.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                  INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:
                                                   Unesite dimenziju niza:
 Unesite elemente niza:
                                                   Unesite elemente niza:
                                                   1.2 3.4 0.0 -5.4 2.1
 -1.1 2.2 3.3
 Zbir elemenata niza je 4.400.
                                                   Zbir elemenata niza je 1.300.
 Proizvod elemenata niza je -7.986
                                                   Proizvod elemenata niza je -0.000.
 Minimalni element niza je -1.100
                                                   Minimalni element niza je -5.400.
Maksimalni element niza je 3.300
                                                   Maksimalni element niza je 3.400.
```

Zadatak 2.3 Korišćenjem pokazivačke sintakse, napisati funkciju koja vrednosti elemenata u prvoj polovini niza povećava za jedan, a u drugoj polovini smanjuje za jedan. Ukoliko niz ima neparan broj elemenata, onda vrednost srednjeg elementa niza ostaviti nepromenjenim. Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju  $n \ (0 < n \le 100)$  celobrojnog niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene napisane funkcije nad učitanim nizom.

```
Primer 1
                                                   Primer 2
                                                INTERAKCIJA SA PROGRAMOM:
INTERAKCIJA SA PROGRAMOM:
  Unesite dimenziju niza:
                                                   Unesite dimenziju niza:
  Unesite elemente niza:
                                                   Unesite elemente niza:
  12345
                                                   4 -3 2 -1
 Transformisan niz je:
                                                   Transformisan niz je:
 2 3 3 3 4
                                                  5 -2 1 -2
 Primer 3
                                                   Primer 4
INTERAKCIJA SA PROGRAMOM:
                                                INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza: 0
                                                   Unesite dimenziju niza: 101
IZLAZ ZA GREŠKE:
                                                  IZLAZ ZA GREŠKE:
 Greska: Neodgovarajuca dimenzija niza.
                                                   Greska: Neodgovarajuca dimenzija niza.
```

Zadatak 2.4 Napisati program koji ispisuje broj prihvaćenih argumenata komandne linije, a zatim i same argumenate kojima prethode njihovi redni brojevi. Nakon toga ispisati prve karaktere svakog od argumenata. Zadatak rešiti:

(a) korišćenjem indeksne sintakse,

#### (b) korišćenjem pokazivačke sintakse.

Od korisnika sa ulaza tražiti da izabere koje od ova dva rešenja treba koristiti prilikom ispisa.

#### Primer 1 Primer 2 POKRETANJE: ./a.out prvi 2. treci -4 POKRETANJE: ./a.out INTERAKCIJA SA PROGRAMOM: INTERAKCIJA SA PROGRAMOM: Broj argumenata komandne linije je 5. Broj argumenata komandne linije je 1. Kako zelite da ispisete argumente, Kako zelite da ispisete argumente, koriscenjem indeksne ili pokazivacke koriscenjem indeksne ili pokazivacke sintakse (I ili P)? Isintakse (I ili P)? P Argumenti komandne linije su: Argumenti komandne linije su: 0 ./a.out 0 ./a.out 1 prvi Pocetna slova argumenata komandne 2 2. linije: 3 treci 4 -4 Pocetna slova argumenata komandne . p 2 t -

Zadatak 2.5 Korišćenjem pokazivačke sintakse, napisati funkciju koja za datu nisku ispituje da li je palindrom. Napisati program koji vrši prebrojavanje argumenata komandne linije koji su palindromi.

```
Primer 1

Pokretanje: ./a.out a b 11 212

Interakcija sa programom:
Broj argumenata
koji su palindromi je 4.

Pokretanje: ./a.out

Interakcija sa programom:
Broj argumenata
koji su palindromi je 0.
```

**Zadatak 2.6** Napisati program koji kao prvi argument komandne linije prihvata putanju do datoteke za koju treba proveriti koliko reči ima n karaktera, gde se n zadaje kao drugi argument komandne linije. Smatrati da reč ne sadrži više od 100 karaktera. U zadatku ne koristiti ugrađene funkcije za rad sa niskama, već implementirati svoje koristeći pokazivačku sintaksu.

```
POKRETANJE: ./a.out ulaz.txt

ULAZ.TXT

Ovo je sadrzaj datoteke i u njoj ima reci koje imaju 1 karakter

INTERAKCIJA SA PROGRAMOM:
IZLAZ ZA GREŠKE:
Greska: Nedovoljan broj argumenata komandne linije.
Program se poziva sa
./a.out ime_dat br_karaktera
```

Zadatak 2.7 Napisati program koji kao prvi argument komandne linije prihvata putanju do datoteke za koju treba proveriti koliko reči ima zadati sufiks (ili prefiks), koji se zadaje kao drugi argument komandne linije. Smatrati da reč ne sadrži više od 100 karaktera. Program je neophodno pozvati sa jednom od opcija -s ili -p u zavisnosti od čega treba proveriti koliko reči ima zadati sufiks (ili prefiks). U zadatku ne koristiti ugrađene funkcije za rad sa niskama, već implementirati svoje koristeći pokazivačku sintaksu.

#### Primer 1

```
POKRETANJE: ./a.out ulaz.txt ke -s

ULAZ.TXT

Ovo je sadrzaj datoteke i u njoj ima
reci koje se zavrsavaju na ke

INTERAKCIJA SA PROGRAMOM:
Broj reci koje se zavrsavaju na ke je 2.
```

#### Primer 3

```
POKRETANJE: ./a.out ulaz.txt sa -p

DATOTEKA ULAZ.TXT NE POSTOJI

INTERAKCIJA SA PROGRAMOM:
IZLAZ ZA GREŠKE:
Greska: Neuspesno otvaranje
datoteke ulaz.txt.
```

#### Primer 2

```
POKRETANJE: ./a.out ulaz.txt sa -p

ULAZ.TXT

Ovo je sadrzaj datoteke i u njoj ima
reci koje pocinju sa sa

INTERAKCIJA SA PROGRAMOM:
Broj reci koje pocinju na sa je 3.
```

#### Primer 4

```
POKRETANJE: ./a.out ulaz.txt

ULAZ.TXT

Ovo je sadrzaj ulaza.

INTERAKCIJA SA PROGRAMOM:
IZLAZ ZA GREŠKE:
Greska: Nedovoljan broj argumenata komandne linije.
Program se poziva sa
./a.out ime_dat suf/pref -s/-p
```

## 2.2 Višedimenzioni nizovi

**Zadatak 2.8** Data je kvadratna matrica dimenzije  $n \times n$ .

- (a) Napisati funkciju koja izračunava trag matrice (sumu elemenata na glavnoj dijagonali).
- (b) Napisati funkciju koja izračunava euklidsku normu matrice (koren sume kvadrata svih elemenata).
- (c) Napisati funkciju koja izračunava gornju vandijagonalnu normu matrice (sumu apsolutnih vrednosti elemenata iznad glavne dijagonale).

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati broj vrsta (ili kolona) kvadratne matrice n ( $0 < n \le 100$ ), a zatim i elemente matrice. Na standardni izlaz ispisati učitanu matricu, a zatim trag, euklidsku normu i vandijagonalnu normu učitane matrice.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                  INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
                                                   Unesite dimenziju matrice:
 Unesite elemente matrice po vrstama:
                                                  IZLAZ ZA GREŠKE:
 1 -2 3
                                                   Greska: Neodgovarajuca dimenzija
 4 -5 6
                                                   matrice.
 7 -8 9
 Trag matrice je 5.
 Euklidska norma matrice je 16.88.
 Vandijagonalna norma matrice je 11.
```

**Zadatak 2.9** Date su dve kvadratne matrice istih dimenzija  $n \times n$ .

- (a) Napisati funkciju koja proverava da li su matrice jednake.
- (b) Napisati funkciju koja izračunava zbir matrica.
- (c) Napisati funkciju koja izračunava proizvod matrica.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati broj vrsta kvadratnih matrica n ( $0 < n \le 100$ ), a zatim i elemente matrica. Na standardni izlaz ispisati da li su matrice jednake, a zatim ispisati zbir i proizvod učitanih matrica.

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrica: 3
 Unesite elemente prve matrice po vrstama:
 123
 123
 123
 Unesite elemente druge matrice po vrstama:
 123
 Matrice su jednake.
 Zbir matrica je:
 2 4 6
 2 4 6
 Proizvod matrica je:
 6 12 8
 6 12 8
 6 12 8
```

**Zadatak 2.10** 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 matrice nalazi broj 1, a nije u relaciji ukoliko se tu nalazi broj 0.

- (a) Napisati funkciju koja proverava da li je relacija zadata matricom refleksivna.
- (b) Napisati funkciju koja proverava da li je relacija zadata matricom simetrična
- (c) Napisati funkciju koja proverava da li je relacija zadata matricom tranzitivna.
- (d) Napisati funkciju koja određuje refleksivno zatvorenje relacije (najmanju refleksivnu relaciju koja je nadskup date).
- (e) Napisati funkciju koja određuje simetrično zatvorenje relacije (najmanju simetričnu relaciju koja je nadskup date).
- (f) Napisati funkciju koja određuje refleksivno-tranzitivno zatvorenje relacije (najmanju refleksivnu i tranzitivnu relaciju koja sadrži datu). NAPOMENA: Koristiti Varšalov algoritam.

Napisati program koji učitava matricu iz datoteke čije se ime zadaje kao prvi argument komandne linije. U prvoj liniji datoteke nalazi se broj vrsta kvadratne matrice n (0 <  $n \le 64$ ), a potom i sami elementi matrice. Na standardni izlaz ispisati rezultat testiranja napisanih funkcija.

```
POKRETANJE: ./a.out ulaz.txt
ULAZ.TXT
 1 0 0 0
 0 1 1 0
 0 0 1 0
  0 0 0 0
INTERAKCIJA SA PROGRAMOM:
 Relacija nije refleksivna.
 Relacija nije simetricna.
  Relacija jeste tranzitivna.
 Refleksivno zatvorenje relacije:
  1 0 0 0
  0 1 1 0
 0 0 1 0
  0 0 0 1
  Simetricno zatvorenje relacije:
  1 0 0 0
  0 1 1 0
 0 1 1 0
 0 0 0 0
  Refleksivno-tranzitivno zatvorenje relacije:
 1 0 0 0
 0 1 1 0
 0 0 1 0
 0 0 0 1
```

**Zadatak 2.11** Data je kvadratna matrica dimenzije  $n \times n$ .

- (a) Napisati funkciju koja određuje najveći element matrice na sporednoj dijagonali.
- (b) Napisati funkciju koja određuje indeks kolone koja sadrži najmanji element matrice.
- (c) Napisati funkciju koja određuje indeks vrste koja sadrži najveći element matrice.
- (d) Napisati funkciju koja određuje broj negativnih elemenata matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati elemente celobrojne kvadratne matrice čiji se broj vrsta  $n\ (0 < n \le 32)$  zadaje kao argument komandne linije. Na standardni izlaz ispisati rezultat primene prethodno napisanih funkcija.

```
POKRETANJE: ./a.out 3

INTERAKCIJA SA PROGRAMOM:
Unesite elemente matrice dimenzije 3x3:
1 2 3
-4 -5 -6
7 8 9

Najveci element sporedne dijagonale je 7.
Indeks kolone sa najmanjim elementom je 2.
Indeks vrste sa najvecim elementom je 2.
Broj negativnih elemenata matrice je 3.
```

#### Primer 2

```
POKRETANJE: ./a.out 4

INTERAKCIJA SA PROGRAMOM:
Unesite elemente matrice dimenzije 4x4:
-1 -2 -3 -4
-5 -6 -7 -8
-9 -10 -11 -12
-13 -14 -15 -16

Najveci element sporedne dijagonale je -4.
Indeks kolone sa najmanjim elementom je 3.
Indeks vrste sa najvecim elementom je 0.
Broj negativnih elemenata matrice je 16.
```

**Zadatak 2.12** Napisati funkciju kojom se proverava da li je zadata kvadratna matrica dimenzije  $n \times n$  ortonormirana. Matrica je ortonormirana ako je skalarni proizvod svakog para različitih vrsta jednak nuli, a skalarni proizvod vrste sa samom sobom jednak jedinici. Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati broj vrsta celobrojne kvadratne matrice  $n \ (0 < n \le 32)$ , a zatim i njene elemente. Na standardni izlaz ispisati rezultat primene napisane funkcije na učitanu matricu.

```
Primer 1
                                                 Primer 2
INTERAKCIJA SA PROGRAMOM:
                                               INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 4
                                                 Unesite broj vrsta matrice: 3
 Unesite elemente matrice po vrstama:
                                                 Unesite elemente matrice po vrstama:
 1000
                                                 123
 0 1 0 0
                                                 567
 0 0 1 0
                                                 142
 0001
                                                 Matrica nije ortonormirana.
 Matrica je ortonormirana.
```

**Zadatak 2.13** Data je matrica dimenzije  $n \times m$ .

(a) Napisati funkciju koja učitava elemente matrice sa standardnog ulaza

(b) Napisati funkciju koja na standardni izlaz spiralno ispisuje elemente matrice, u smeru kretanja kazaljke na satu.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati broj vrsta n ( $0 < n \le 10$ ) i broj kolona m ( $0 < n \le 10$ ) matrice, a zatim i njene elemente. Na standardni izlaz spiralno ispisati elemente učitane matrice.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                  INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta i broj kolona
                                                   Unesite broj vrsta i broj kolona
 matrice:
                                                   matrice:
 33
                                                   34
 Unesite elemente matrice po vrstama:
                                                   Unesite elemente matrice po vrstama:
 123
                                                   1234
 456
789
                                                   5 6 7 8
                                                   9 10 11 12
 Spiralno ispisana matrica:
                                                   Spiralno ispisana matrica:
 1 2 3 6 9 8 7 4 5
                                                   1 2 3 4 8 12 11 10 9 5 6 7
```

Zadatak 2.14 Napisati funkciju koja izračunava k-ti stepen kvadratne matrice dimenzije  $n \times n$  ( $0 < n \le 32$ ). Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati broj vrsta celobrojne matrice n, elemente matrice i stepen k ( $0 < k \le 10$ ). Na standardni izlaz ispisati rezultat primene napisane funkcije. Napomena: Voditi računa da se prilikom stepenovanja matrice izvrši što manji broj množenja.

# 2.3 Dinamička alokacija memorije

Zadatak 2.15 Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva, a zatim i njegove elemente. Ne praviti nikakve pretpostavke o dimenziji niza. Na standardni izlaz ispisati ove brojeve u obrnutom poretku.

```
INTERAKCIJA SA PROGRAMOM:
Unesite dimenziju niza: 3
Unesite elemente niza: 1 -2 3
Niz u obrnutom poretku je: 3 -2 1
```

#### Primer 2

```
|| INTERAKCIJA SA PROGRAMOM:
| Unesite dimenziju niza: -1
| Greska: Neuspesna alokacija memorije.
```

Zadatak 2.16 Napisati program koji sa standardnog ulaza učitava niz celih brojeva. Brojevi se unose sve dok se ne unese nula. Ne praviti nikakve pretpostavke o dimenziji niza. Na standardni izlaz ispisati ovaj niz brojeva u obrnutom poretku. Zadatak uraditi na dva načina:

- (a) realokaciju memorije niza vršiti korišćenjem malloc() funkcije,
- (b) realokaciju memorije niza vršiti korišćenjem realloc() funkcije.

Od korisnika sa ulaza tražiti da izabere način realokacije memorije.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite zeljeni nacin realokacije
(M ili R):
M
Unesite brojeve, nulu za kraj:
1 -2 3 -4 0
Niz u obrnutom poretku je:
-4 3 -2 1
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite zeljeni nacin realokacije
(M ili R):
R
Unesite brojeve, nulu za kraj:
6-15-24-30
Niz u obrnutom poretku je:
34-25-16
```

Zadatak 2.17 Napisati funkciju koja kao rezultat vraća nisku koja se dobija nadovezivanjem dve niske, bez promene njihovog sadržaja. Napisati program koji testira rad napisane funkcije. Sa standardnog ulaza učitati dve niske karaktera (pretpostaviti da niske nisu duže od 50 karaktera i da ne sadrže praznine). Na standardni izlaz ispisati nisku koja se dobija njihovim nadovezivanjem. Za rezultujuću nisku dinamički alocirati memoriju.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite dve niske karaktera:
Jedan Dva
Nadovezane niske: JedanDva
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite dve niske karaktera:
Ana Marija
Nadovezane niske: AnaMarija
```

**Zadatak 2.18** Napisati program koji sa standardnog ulaza učitava matricu realnih brojeva. Prvo se učitavaju broj vrsta n i broj kolona m matrice (ne praviti nikakve pretpostavke o njihovoj veličini), a zatim i elementi matrice. Na standardni izlaz ispisati trag matrice.

```
INTERAKCIJA SA PROGRAMOM:

Unesite broj vrsta i broj kolona:
2 3

Unesite elemente matrice po vrstama:
1.2 2.3 3.4
4.5 5.6 6.7

Trag unete matrice je 6.80.
```

#### Primer 2

```
| INTERAKCIJA SA PROGRAMOM:

Unesite broj vrsta i broj kolona:

2 2

Unesite elemente matrice po vrstama:

-0.1 -0.2

-0.3 -0.4

Trag unete matrice je -0.50.
```

Zadatak 2.19 Napisati biblioteku za rad sa celobrojnim matricama.

- (a) Napisati funkciju int \*\*alociraj\_matricu(int n, int m) koja dinamički alocira memoriju potrebnu za matricu dimenzije  $n \times m$ .
- (b) Napisati funkciju int \*\*alociraj\_kvadratnu\_matricu(int n) koja alocira memoriju za kvadratnu matricu dimenzije n.
- (c) Napisati funkciju int \*\*dealociraj\_matricu(int \*\*A, int n) koja dealocira memoriju matrice sa n vrsta. Povratna vrednost ove funkcije treba da bude "prazna" matrica.
- (d) Napisati funkciju void ucitaj\_matricu(int \*\*A, int n, int m) koja učitava već alociranu matricu dimenzije  $n \times m$  sa standardnog ulaza.
- (e) Napisati funkciju void ucitaj\_kvadratnu\_matricu(int \*\*A, int n) koja učitava već alociranu kvadratnu matricu dimenzije  $n \times n$  sa standardnog ulaza.
- (f) Napisati funkciju void ispisi\_matricu(int \*\*A, int n, int m) koja ispisuje matricu dimenzije  $n \times m$  na standardnom izlazu.
- (g) Napisati funkciju void ispisi\_kvadratnu\_matricu(int \*\*A, int n) koja ispisuje kvadratnu matricu dimenzije  $n \times n$  na standardnom izlazu.
- (h) Napisati funkciju int ucitaj\_matricu\_iz\_datoteke(int \*\*A, int n, int m, FILE \* f) koja učitava već alociranu matricu dimenzije  $n \times m$  iz već otvorene datoteke f. U slučaju neuspešnog učitavanja vratiti vrednost različitu od 0.
- (i) Napisati funkciju int ucitaj\_kvadratnu\_matricu\_iz\_datoteke(int \*\*A, int n, FILE \* f) koja učitava već alociranu kvadratnu matricu dimenzije n × n iz već otvorene datoteke f. U slučaju neuspešnog učitavanja vratiti vrednost različitu od 0.

- (j) Napisati funkciju int upisi\_matricu\_u\_datoteku(int \*\*A, int n, int m, FILE \* f) koja upisuje matricu dimenzije n×m u već otvorenu datoteku f. U slučaju neuspešnog upisivanja vratiti vrednost različitu od 0.
- (k) Napisati funkciju int upisi\_kvadratnu\_matricu\_u\_datoteku(int \*\*A, int n, FILE \* f) koja upisuje kvadratnu matricu dimenzije n × n u već otvorenu datoteku f. U slučaju neuspešnog upisivanja vratiti vrednost različitu od 0.

Napisati programe koji testiraju napisanu biblioteku.

(1) Program učitava dimenziju nekvadratne matrice sa standardnog ulaza, a zatim i samu matricu. Potom, matricu upisati u datoteku matrica.txt.

```
Primer 1
                                                  Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
                                                  Unesite broj vrsta matrice: 5
 Unesite broj kolona matrice: 4
                                                  Unesite broj kolona matrice: 0
 Unesite elemente matrice po vrstama:
                                                 IZLAZ ZA GREŠKE:
 1231
                                                  Greska: Broj vrsta i broj kolona ne mogu
 5678
                                                  biti negativni brojevi.
 9 10 11 12
MATRICA.TXT
 1 2 3 4
 5 6 7 8
 9 10 11 12
```

(2) Program prima kao prvi argument komandne linije putanju do datoteke u kojoj se redom nalazi dimenzija kvadratne matrice i sama matrica, koju treba ispisati na standardnom izlazu.

```
Test 1
                               Test 2
                                                               Test 3
POKRETANJE: ./a.out ulaz.txt || POKRETANJE: ./a.out ulaz.txt || POKRETANJE: ./a.out
ULAZ.TXT
                               ULAZ.TXT
 4
                                dimenzija: 4
                                                               Koriscenje programa:
 1 2 3 4
                                 1 2 3 4
                                                                ./a.out datoteka
 5 6 7 8
                                5 6 7 8
 9 10 11 12
                                9 10 11 12
 13 14 15 16
                                13 14 15 16
Izlaz:
                               Izlaz za greške:
                                Greska: Neispravan pocetak
 1 2 3 4
 5 6 7 8
                                    datoteke.
 9 10 11 12
 13 14 15 16
```

Zadatak 2.20 Data je celobrojna matrica dimenzije  $n \times m$ . Napisati funkciju koja ispisuje elemente ispod glavne dijagonale matrice (uključujući i glavnu dijagonalu). Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati n i m (ne praviti nikakve pretpostavke o njihovoj veličini), zatim učitati elemente matrice i na standardni izlaz ispisati elemente ispod glavne dijagonale matrice. Napomena: Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj vrsta i broj kolona:
2 3
Unesite elemente matrice po vrstama:
1 -2 3
-4 5 -6
Elementi ispod glavne dijagonale matrice:
1
-4 5
```

Zadatak 2.21 Za zadatu matricu dimenzije  $n \times m$  napisati funkciju koja izračunava redni broj kolone matrice čiji je zbir maksimalan. Napisati program koji testira ovu funkciju. Sa standardnog ulaza učitati dimenzije matrice n i m (ne praviti nikakve pretpostavke o njihovoj veličini), a zatim elemente matrice. Na standardni izlaz ispisati redni broj kolone matrice sa maksimalnim zbirom. Ukoliko ima više takvih, ispisati prvu. Napomena: Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj vrsta i broj kolona:
2 3
Unesite elemente matrice po vrstama:
1 2 3
4 5 6
Kolona pod rednim brojem 3 ima
najveci zbir.
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj vrsta i broj kolona:
2 4
Unesite elemente matrice po vrstama:
1 2 3 4
8 7 6 5
Kolona pod rednim brojem 1 ima
naiveci zbir.
```

**Zadatak 2.22** Data je realna kvadratna matrica dimenzije  $n \times n$ .

- (a) Napisati funkciju koja izračunava zbir apsolutnih vrednosti matrice ispod sporedne dijagonale.
- (b) Napisati funkciju koja menja sadržaj matrice tako što polovi elemente iznad glavne dijagonale, duplira elemente ispod glavne dijagonale, dok elemente na glavnoj dijagonali ostavlja nepromenjene.

Napisati program koji testira ove funkcije za matricu koja se učitava iz datoteke čije se ime zadaje kao argument komandne linije. U datoteci se nalazi prvo dimenzija matrice, a zatim redom elementi matrice.

```
        Primer 1
        INTERAKCIJA SA PROGRAMOM:

        POKRETANJE: ./a.out matrica.txt
        Zbir apsolutnih vrednosti ispod sporedne dijagonale je 25.30.

        MATRICA.TXT
        Transformisana matrica je:

        3
        1.10 -1.10 1.65

        1.1 -2.2 3.3
        -8.80 5.50 -3.30

        -4.4 5.5 -6.6
        15.40 -17.60 9.90

        7.7 -8.8 9.9
        15.40 -17.60 9.90
```

Zadatak 2.23 Napisati program koji na osnovu dve realne matrice dimenzije  $m \times n$  formira matricu dimenzije  $2 \cdot m \times n$  tako što naizmenično kombinuje jednu vrstu prve matrice i jednu vrstu druge matrice. Matrice su zapisane u datoteci matrice.txt. U prvom redu datoteke se nalaze broj vrsta m i broj kolona n matrica, u narednih m redova se nalaze vrste prve matrice, a u narednih m redova vrste druge matrice. Rezultujuću matricu ispisati na standardni izlaz.

```
Primer 1
                                                   INTERAKCIJA SA PROGRAMOM:
                                                    1.1 -2.2 3.3
POKRETANJE: ./a.out matrice.txt
                                                    -1.1 2.2 -3.3
                                                    -4.4 5.5 -6.6
MATRICE.TXT
                                                    4.4 -5.5 6.6
 3
                                                    7.7 -8.8 9.9
 1.1 -2.2 3.3
                                                    -7.7 8.8 -9.9
 -4.4 5.5 -6.6
 7.7 -8.8 9.9
 -1.1 2.2 -3.3
 4.4 -5.5 6.6
 -7.7 8.8 -9.9
```

Zadatak 2.24 Na ulazu se zadaje niz celih brojeva čiji se unos završava nulom. Napisati funkciju koja od zadatog niza formira matricu tako da prva vrsta odgovara unetom nizu, a svaka naredna se dobija cikličkim pomeranjem elemenata niza za jednu poziciju ulevo. Napisati program koji testira ovu funkciju. Rezultujuću matricu ispisati na standardni izlaz. NAPOMENA: Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                 INTERAKCIJA SA PROGRAMOM:
 Unesite elemente niza, nulu za kraj:
                                                   Unesite elemente niza, nulu za kraj:
 1230
                                                   -5 -2 -4 -1 0
 Trazena matrica je:
                                                  Trazena matrica je:
 1 2 3
                                                   -5 -2 -4 -1
 2 3 1
                                                   -2 -4 -1 -5
 3 1 2
                                                   -4 -1 -5 -2
                                                   -1 -5 -2 -4
```

Zadatak 2.25 Petar sakuplja sličice igrača za predstojeće Svetsko prvenstvo u fudbalu. U datoteci slicice.txt se nalaze informacije o sličicama koje mu nedostaju u formatu:

redni\_broj\_sličice ime\_reprezentacije\_kojoj\_sličica\_pripada Pomozite Petru da otkrije koliko mu sličica ukupno nedostaje, kao i da pronađe ime reprezentacije čijih sličica ima najmanje. Dobijene podatke ispisati na standardni izlaz. Napomena: Koristiti realloc() funkciju za realokaciju memorije.

# Primer 1 | SLICICE.TXT Petru ukupno nedostaje 7 slicica. | 3 Brazil Reprezentacija za koju je sakupio | | 6 Nemacka najmanji broj slicica je Brazil. | 2 Kamerun | | 1 Brazil | | 2 Engleska | | 4 Engleska | | 5 Brazil

\* Zadatak 2.26 U datoteci temena.txt se nalaze tačke koje predstavljaju temena nekog n-tougla. Napisati program koji na osnovu sadržaja datoteke na standardni izlaz ispisuje o kom n-touglu je reč, a zatim i vrednosti njegovog obima i površine. Pretpostavka je da će mnogougao biti konveksan.

```
Primer 1
                                                   INTERAKCIJA SA PROGRAMOM:
                                                    U datoteci su zadata temena
TEMENA.TXT
 -1 -1
                                                     cetvorougla.
                                                     Obim je 8.
 1 -1
                                                     Povrsina je 4.
 1 1
 -1 1
 Primer 2
                                                   INTERAKCIJA SA PROGRAMOM:
                                                    U datoteci su zadata temena
TEMENA.TXT
 -1.75 -1.5
                                                     petougla.
                                                     Obim je 18.80.
 3 1.5
                                                     Povrsina je 22.59.
 2.2 3.1
 -2 4
 -4.1 1
```

# 2.4 Pokazivači na funkcije

**Zadatak 2.27** Napisati program koji tabelarno štampa vrednosti proizvoljne realne funkcije sa jednim realnim argumentomo, odnosno izračunava i ispisuje vrednosti date funkcije u n ekvidistantnih tačaka na intervalu [a,b]. Re-

alni brojevi a i b (a < b) kao i ceo broj n ( $n \ge 2$ ) se učitavaju sa standardnog ulaza. Ime funkcije se zadaje kao argument komandne linije (sin, cos, tan, atan, acos, asin, exp, log, log10, sqrt, floor, ceil, sqr).

```
Primer 1
                                                    Primer 2
POKRETANJE: ./a.out sin
                                                   POKRETANJE: ./a.out cos
INTERAKCIJA SA PROGRAMOM:
                                                   INTERAKCIJA SA PROGRAMOM:
 Unesite krajeve intervala:
                                                    Unesite krajeve intervala:
 Koliko tacaka ima na ekvidistantnoj
                                                    Koliko tacaka ima na ekvidistantnoj
 mrezi (ukljucujuci krajeve intervala)?
                                                    mrezi (ukljucujuci krajeve intervala)?
   -0.50000 | -0.47943 |
                                                     | 0.00000 | 1.00000 |
   0.00000 | 0.00000 |
                                                     | 0.66667 | 0.78589 |
 I 0.50000 I 0.47943
                                                     I 1.33333 I 0.23524 I
 | 1.00000 | 0.84147 |
                                                     | 2.00000 | -0.41615 |
```

**Zadatak 2.28** Napisati funkciju koja izračunava limes funkcije f(x) u tački a. Adresa funkcije f čiji se limes računa se prenosi kao parametar funkciji za računanje limesa. Limes se računa sledećom aproksimacijom (vrednosti n i a uneti sa standardnog ulaza kao i ime funkcije):

$$\lim_{x\to a} f(x) = \lim_{n\to\infty} f(a + \frac{1}{n})$$

```
Primer 1

| Interakcija sa programom:
| Unesite ime funkcije, n i a:
| tan 10000 1.570795
| Limes funkcije tan je -10134.46.
| Primer 2

| Interakcija sa programom:
| Unesite ime funkcije, n i a:
| cos 5000 0.25
| Limes funkcije cos je 0.97.
```

**Zadatak 2.29** Napisati funkciju koja određuje integral funkcije f(x) na intervalu [a, b]. Adresa funkcije f se prenosi kao parametar. Integral se računa prema formuli:

$$\int_{a}^{b} f(x) = h \cdot \left( \frac{f(a) + f(b)}{2} + \sum_{i=1}^{n} f(a + i \cdot h) \right)$$

Vrednost h se izračunava po formuli h=(b-a)/n, dok se vrednosti n, a i b unose sa standardnog ulaza kao i ime funkcije iz zaglavlja math.h. Na standardni izlaz ispisati vrednost integrala.

# Primer 1 | INTERAKCIJA SA PROGRAMOM: | Unesite ime funkcije, n, a i b: | cos 6000 -1.5 3.5 | Sin 10000 -5.2 2.1 | | Vrednost integrala je 0.645931.

**Zadatak 2.30** Napisati funkciju koja približno izračunava integral funkcije f(x) na intervalu [a,b]. Funkcija f se prosleđuje kao parametar, a integral se procenjuje po Simpsonovoj formuli:

$$I = \frac{h}{3} \left( f(a) + 4 \sum_{i=1}^{n/2} f(a + (2i - 1)h) + 2 \sum_{i=1}^{n/2 - 1} f(a + 2ih) + f(b) \right)$$

Granice intervala i n su argumenti funkcije. Napisati program, koji kao argumente komandne linije prihvata ime funkcije iz zaglavlja  $\mathtt{math.h}$ , krajeve intervala i n, a na standardni izlaz ispisuje vrednost odgovarajućeg integrala.

```
Primer 1

| Interakcija sa programom:
| Unesite ime funkcije, n, a i b:
| sin 100 -1.0 3.0 | Unesite ime funkcije, n, a i b:
| tan 5000 -4.1 -2.3 | Uneste integrala je 1.530295. | Uneste integrala je -0.147640.
```

# 2.5 Rešenja

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

#define MAX 100

6  /* Funkcija obrce elemente niza koriscenjem indekse sintakse */
void obrni_niz_v1(int a[], int n)
{
   int i, j;

for (i = 0, j = n - 1; i < j; i++, j--) {
   int t = a[i];
   a[i] = a[j];
   a[j] = t;
}</pre>
```

```
16 }
18 /* Funkcija obrce elemente niza koriscenjem pokazivacke sintakse */
  void obrni_niz_v2(int *a, int n)
20 | {
    /* Pokazivaci na elemente niza */
   int *prvi, *poslednji;
   /* Vrsi se obrtanje niza */
   for (prvi = a, poslednji = a + n - 1; prvi < poslednji;) {
     int t = *prvi;
26
     /* Na adresu na koju pokazuje pokazivac "prvi" postavlja se
28
         vrednost koja se nalazi na adresi na koju pokazuje pokazivac
         "poslednji". Nakon toga se pokazivac "prvi" uvecava za jedan
30
        sto za posledicu ima da "prvi" pokazuje na sledeci element u
        nizu */
      *prvi++ = *poslednji;
34
      /* Vrednost promenljive "t" se postavlja na adresu na koju
         pokazuje pokazivac "poslednji". Ovaj pokazivac se zatim
36
         umanjuje za jedan, cime pokazivac "poslednji" pokazuje na
         element koji mu prethodi u nizu */
38
      *poslednji-- = t;
40
    42
     Drugi nacin za obrtanje niza
44
      for (prvi = a, poslednji = a + n - 1; prvi < poslednji;</pre>
                                          prvi++, poslednji--) {
46
       int t = *prvi;
       *prvi = *poslednji;
48
        *poslednji = t;
     ********************
52 }
54 int main()
    /* Deklarise se niz od najvise MAX elemenata */
56
    int a[MAX];
   /* Broj elemenata niza a */
   int n;
    /* Pokazivac na elemente niza */
    int *p;
64
    printf("Unesite dimenziju niza: ");
    scanf("%d", &n);
66
```

```
/* Proverava se da li je doslo do prekoracenja ogranicenja
68
       dimenzije */
    if (n \le 0 | | n > MAX) {
      fprintf(stderr, "Greska: Neodgovarajuca dimenzija niza.\n");
      exit(EXIT_FAILURE);
72
74
    printf("Unesite elemente niza:\n");
    for (p = a; p - a < n; p++)
      scanf("%d", p);
78
    obrni_niz_v1(a, n);
80
    printf("Nakon obrtanja elemenata, niz je:\n");
82
    for (p = a; p - a < n; p++)
      printf("%d ", *p);
84
    printf("\n");
86
    obrni_niz_v2(a, n);
88
    printf("Nakon ponovnog obrtanja elemenata, niz je:\n");
90
    for (p = a; p - a < n; p++)
      printf("%d ", *p);
92
    printf("\n");
94
     exit(EXIT_SUCCESS);
  }
96
```

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

#define MAX 100

/* Funkcija izracunava zbir elemenata niza */
double zbir(double *a, int n)

{
    double s = 0;
    int i;

for (i = 0; i < n; s += *(a + i++));

return s;
}

/* Funkcija izracunava proizvod elemenata niza */
double proizvod(double *a, int n)</pre>
```

```
double p = 1;
    for (; n; n--)
     p *= (*(a + n - 1));
24
    return p;
26 }
28 /* Funkcija izracunava minimalni element niza */
  double min(double *a, int n)
30 {
    /* Na pocetku, minimalni element je prvi element */
   double min = *a;
    int i;
34
    /* Ispituje se da li se medju ostalim elementima niza nalazi
      minimalni */
36
   for (i = 1; i < n; i++)
     if (*(a + i) < min)
38
        min = *(a + i);
40
    return min;
42 }
44 /* Funkcija izracunava maksimalni element niza */
  double max(double *a, int n)
46 {
    /* Na pocetku, maksimalni element je prvi element */
    double max = *a;
48
    /* Ispituje se da li se medju ostalim elementima niza nalazi
50
       maksimalni */
   for (a++, n--; n > 0; a++, n--)
     if (*a > max)
        max = *a;
54
    return max;
58
60 int main()
    double a[MAX];
62
    int n, i;
64
    printf("Unesite dimenziju niza: ");
    scanf("%d", &n);
66
    /* Proverava se da li je doslo do prekoracenja ogranicenja
68
       dimenzije */
    if (n \le 0 | | n > MAX) {
70
```

```
fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
      exit(EXIT FAILURE);
74
    printf("Unesite elemente niza:\n");
    for (i = 0; i < n; i++)
76
      scanf("%lf", a + i);
78
    /* Vrsi se testiranje definisanih funkcija */
    printf("Zbir elemenata niza je %5.3f.\n", zbir(a, n));
80
    printf("Proizvod elemenata niza je %5.3f.\n", proizvod(a, n));
    printf("Minimalni element niza je %5.3f.\n", min(a, n));
82
    printf("Maksimalni element niza je %5.3f.\n", max(a, n));
84
    exit(EXIT_SUCCESS);
  }
86
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAX 100
  /* Funkcija povecava za jedan sve elemente u prvoj polovini niza a
     smanjuje za jedan sve elemente u drugoj polovini niza. Ukoliko
     niz ima neparan broj elemenata, srednji element ostaje
     nepromenjen */
10 void povecaj_smanji(int *a, int n)
12
    int *prvi = a;
    int *poslednji = a + n - 1;
14
    while (prvi < poslednji) {
16
      /* Uvecava se element na koji pokazuje pokazivac "prvi" */
      (*prvi)++;
18
      /* Pokazivac "prvi" se pomera na sledeci element */
20
      prvi++;
      /* Smanjuje se vrednost elementa na koji pokazuje pokazivac
         "poslednji" */
      (*poslednji)--;
      /* Pokazivac "poslednji" se pomera na prethodni element */
      poslednji--;
28
30
     Drugi nacin:
```

```
while (prvi < poslednji) {</pre>
       (*prvi++)++;
34
       (*poslednji--)--;
36
         *************************
  }
38
40 int main()
    int a[MAX];
42
    int n;
    int *p;
44
    printf("Unesite dimenziju niza: ");
46
    scanf("%d", &n);
48
    /* Proverava se da li je doslo do prekoracenja ogranicenja
      dimenzije */
    if (n \le 0 | | n > MAX) {
     fprintf(stderr, "Greska: Neodgovarajuca dimenzija niza.\n");
      exit(EXIT_FAILURE);
54
    printf("Unesite elemente niza:\n");
56
    for (p = a; p - a < n; p++)
     scanf("%d", p);
58
    povecaj_smanji(a, n);
    printf("Transformisan niz je:\n");
    for (p = a; p - a < n; p++)
printf("%d ", *p);
64
    printf("\n");
    exit(EXIT_SUCCESS);
  }
68
```

```
printf("Argumenti komandne linije su:\n");
14
    if (tip_ispisa == 'I') {
      /* Ispisuju se argumenti komandne linije koriscenjem indeksne
         sintakse */
      for (i = 0; i < argc; i++)
18
        printf("%d %s\n", i, argv[i]);
    } else if (tip_ispisa == 'P') {
20
      /* Ispisuju se argumenti komandne linije koriscenjem pokazivacke
         sintakse */
      i = argc;
      for (; argc > 0; argc--)
24
        printf("%d %s\n", i - argc, *argv++);
26
      /* Nakon ove petlje "argc" je jednako nuli a "argv" pokazuje na
         polje u memoriji koje se nalazi iza poslednjeg argumenta
28
         komandne linije. Kako je u promenljivoj "i" sacuvana vrednost
         broja argumenta komandne linije to sada moze ponovo da se
30
         postavi "argv" da pokazuje na nulti argument komandne linije
      argv = argv - i;
      argc = i;
34
    printf("Pocetna slova argumenata komandne linije:\n");
36
    if (tip_ispisa == 'I') {
      /* koristeci indeksnu sintaksu */
38
      for (i = 0; i < argc; i++)
        printf("%c ", argv[i][0]);
40
      printf("\n");
    } else if (tip_ispisa == 'P') {
42
      /* koristeci pokazivacku sintaksu */
      for (i = 0; i < argc; i++)
44
        printf("%c ", **argv++);
      printf("\n");
46
48
    return 0;
  }
50
```

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

#define MAX 100

/* Funkcija ispituje da li je niska palindrom, odnosno da li se isto
cita spreda i odpozadi */
int palindrom(char *niska)

{
```

```
int i, j;
    for (i = 0, j = strlen(niska) - 1; i < j; i++, j--)
      if (*(niska + i) != *(niska + j))
        return 0;
13
    return 1:
15 }
17 int main(int argc, char **argv)
    int i, n = 0;
19
    /* Nulti argument komandne linije je ime izvrsnog programa */
    for (i = 1; i < argc; i++)
     if (palindrom(*(argv + i)))
        n++;
    printf("Broj argumenata koji su palindromi je %d.\n", n);
    return 0;
29 }
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAX_KARAKTERA 100
  /* Implementacija funkcije strlen() iz standardne biblioteke */
7 int duzina(char *s)
    int i;
   for (i = 0; *(s + i); i++);
    return i;
13
  int main(int argc, char **argv)
15 {
    char rec[MAX_KARAKTERA + 1];
17
   int br = 0, n;
    FILE *in;
19
    /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
       greska */
    if (argc < 3) {
      fprintf(stderr, "Greska: ");
      fprintf(stderr, "Nedovoljan broj argumenata komandne linije.\n");
      fprintf(stderr, "Program se poziva sa %s ime_dat br_karaktera\n",
25
              argv[0]);
      exit(EXIT_FAILURE);
27
```

```
/* Otvara se datoteka sa imenom koje se zadaje kao prvi argument
       komandne linije. */
31
    in = fopen(*(argv + 1), "r");
    if (in == NULL) {
33
      fprintf(stderr, "Greska: ");
      fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
35
      exit(EXIT_FAILURE);
37
    n = atoi(*(argv + 2));
39
    /* Broje se reci cija je duzina jednaka broju zadatom drugim
41
       argumentom komandne linije */
    while (fscanf(in, "%s", rec) != EOF)
43
      if (duzina(rec) == n)
        br++;
45
    printf("Broj reci ciji je broj karaktera %d je %d.\n", n, br);
    /* Zatvara se datoteka */
49
    fclose(in);
    exit(EXIT_SUCCESS);
53 }
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAX_KARAKTERA 100
  /* Implementacija funkcije strcpy() iz standardne biblioteke */
void kopiranje_niske(char *dest, char *src)
  {
   int i;
    for (i = 0; *(src + i); i++)
      *(dest + i) = *(src + i);
  }
13
  /* Implementacija funkcije strcmp() iz standardne biblioteke */
int poredjenje_niski(char *s, char *t)
    for (i = 0; *(s + i) == *(t + i); i++)
      if (*(s + i) == '\0')
19
        return 0;
    return *(s + i) - *(t + i);
  }
23
```

```
|/* Implementacija funkcije strlen() iz standardne biblioteke */
int duzina_niske(char *s)
    int i:
   for (i = 0; *(s + i); i++);
   return i;
  /* Funkcija ispituje da li je niska zadata drugim argumentom
    funkcije sufiks niske zadate prvi argumentom funkcije */
  int sufiks_niske(char *niska, char *sufiks)
35 {
    int duzina_sufiksa = duzina_niske(sufiks);
   int duzina_niske_pom = duzina_niske(niska);
    if (duzina_sufiksa <= duzina_niske_pom &&
        poredjenje_niski(niska + duzina_niske_pom -
                         duzina_sufiksa, sufiks) == 0)
     return 1:
41
    return 0;
43 }
45 /* Funkcija ispituje da li je niska zadata drugim argumentom
     funkcije prefiks niske zadate prvi argumentom funkcije */
47 int prefiks_niske(char *niska, char *prefiks)
    int i;
49
    int duzina_prefiksa = duzina_niske(prefiks);
    int duzina_niske_pom = duzina_niske(niska);
    if (duzina_prefiksa <= duzina_niske_pom) {</pre>
     for (i = 0; i < duzina_prefiksa; i++)</pre>
53
        if (*(prefiks + i) != *(niska + i))
          return 0:
      return 1;
    } else
      return 0;
  }
59
int main(int argc, char **argv)
    /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
       greska */
    if (argc < 4) {
      fprintf(stderr, "Greska: ");
      fprintf(stderr, "Nedovoljan broj argumenata komandne linije.\n");
      fprintf(stderr, "Program se poziva sa\n");
      fprintf(stderr, "%s ime_dat suf/pref -s/-p\n", argv[0]);
      exit(EXIT_FAILURE);
    }
    FILE *in;
73
    int br = 0;
    char rec[MAX_KARAKTERA + 1];
```

```
in = fopen(*(argv + 1), "r");
    if (in == NULL) {
      fprintf(stderr, "Greska: ");
      fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
      exit(EXIT_FAILURE);
81
83
    /* Provera se opcija kojom je pozvan program a zatim se ucitavaju
       reci iz datoteke i broji se koliko njih zadovoljava trazeni
85
       uslov */
    if (!(poredjenje_niski(*(argv + 3), "-s"))) {
87
      while (fscanf(in, "%s", rec) != EOF)
        br += sufiks_niske(rec, *(argv + 2));
89
      printf("Broj reci koje se zavrsavaju na %s je %d.\n",
              *(argv + 2), br);
91
    } else if (!(poredjenje_niski(*(argv + 3), "-p"))) {
      while (fscanf(in, "%s", rec) != EOF)
93
        br += prefiks_niske(rec, *(argv + 2));
      printf("Broj reci koje pocinju na %s je %d.\n", *(argv + 2), br);
95
97
    fclose(in);
99
    exit(EXIT_SUCCESS);
  }
```

```
1 #include <stdio.h>
  #include <math.h>
3 #include <stdlib.h>
5 #define MAX 100
7 /* Funkcija izracunava trag matrice */
  int trag(int M[][MAX], int n)
  {
    int trag = 0, i;
    for (i = 0; i < n; i++)
      trag += M[i][i];
    return trag;
13
  /* Funkcija izracunava euklidsku normu matrice */
double euklidska_norma(int M[][MAX], int n)
    double norma = 0.0;
19
    int i, j;
    for (i = 0; i < n; i++)
```

```
23
      for (j = 0; j < n; j++)
        norma += M[i][j] * M[i][j];
    return sqrt(norma);
27 }
29 /* Funkcija izracunava gornju vandijagonalnu normu matrice */
  int gornja_vandijagonalna_norma(int M[][MAX], int n)
31 {
    int norma = 0;
   int i, j;
   for (i = 0; i < n; i++) {
      for (j = i + 1; j < n; j++)
       norma += abs(M[i][j]);
39
    return norma;
  }
41
43 int main()
    int A[MAX][MAX];
45
    int i, j, n;
47
    printf("Unesite broj vrsta matrice: ");
    scanf("%d", &n);
49
    /* Provera prekoracenja dimenzije matrice */
    if (n > MAX || n <= 0) {
      fprintf(stderr, "Greska: Neodgovarajuca dimenzija matrice.\n");
      exit(EXIT_FAILURE);
    printf("Unesite elemente matrice po vrstama:\n ");
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
59
        scanf("%d", &A[i][j]);
    /* Ispis sadrzaja matrice koriscenjem indeksne sintakse */
    for (i = 0; i < n; i++) {
      /* Ispis elemenata i-te vrste */
      for (j = 0; j < n; j++)
        printf("%d ", A[i][j]);
      printf("\n");
    Ispisuju se elemenati matrice koriscenjem pokazivacke sintakse.
     {\tt Kod\ ovako\ definisane\ matrice,\ elementi\ su\ uzastopno\ smesteni\ u}
     memoriju, kao na traci. To znaci da su svi elementi prve vrste
73
     redom smesteni jedan iza drugog. Odmah iza poslednjeg elementa
```

```
prve vrste smesten je prvi element druge vrste za kojim slede
     svi elementi te vrste i tako dalje redom.
     for( i = 0; i < n; i++) {
       for ( j=0 ; j < n ; j++)
79
         printf("%d ", *(*(A+i)+j));
       printf("\n");
81
83
    /* Ispisuje se rezultat na standardni izlaz */
85
    int tr = trag(A, n);
    printf("Trag matrice je %d.\n", tr);
87
    printf("Euklidska norma matrice je %.2f.\n",
89
            euklidska_norma(A, n));
    printf("Vandijagonalna norma matrice je = %d.\n",
91
           gornja_vandijagonalna_norma(A, n));
93
    exit(EXIT_SUCCESS);
  }
95
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAX 100
  /* Funkcija ucitava elemente kvadratne matrice dimenzije n sa
     standardnog ulaza */
  void ucitaj_matricu(int m[][MAX], int n)
    int i, j;
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
13
        scanf("%d", &m[i][j]);
15 }
  /* Funkcija ispisuje elemente kvadratne matrice dimenzije n na
     standardni izlaz */
  void ispisi_matricu(int m[][MAX], int n)
    int i, j;
    for (i = 0; i < n; i++) {
23
      for (j = 0; j < n; j++)
        printf("%d ", m[i][j]);
      printf("\n");
```

```
| }
29
  /* Funkcija proverava da li su zadate kvadratne matrice a i b
    dimenzije n jednake */
  int jednake_matrice(int a[][MAX], int b[][MAX], int n)
33 {
    int i, j;
35
    for (i = 0; i < n; i++)
     for (j = 0; j < n; j++)
        if (a[i][j] != b[i][j])
          return 0;
39
    /* Prosla je provera jednakosti za sve parove elemenata koji su na
       istim pozicijama. To znaci da su matrice jednake */
    return 1;
43
45
  /* Funkcija izracunava zbir dve kvadratne matice */
47 void saberi(int a[][MAX], int b[][MAX], int c[][MAX], int n)
    int i, j;
49
   for (i = 0; i < n; i++)
     for (j = 0; j < n; j++)
        c[i][j] = a[i][j] + b[i][j];
  }
  /* Funkcija izracunava proizvod dve kvadratne matice */
void pomnozi(int a[][MAX], int b[][MAX], int c[][MAX], int n)
    int i, j, k;
59
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++) {
        /* Mnozi se i-ta vrsta prve sa j-tom kolonom druge matrice */
        c[i][j] = 0;
        for (k = 0; k < n; k++)
          c[i][j] += a[i][k] * b[k][j];
  7
  int main()
71 {
    /* Matrice ciji se elementi zadaju sa ulaza */
    int a[MAX][MAX], b[MAX][MAX];
    /* Matrice zbira i proizvoda */
    int zbir[MAX][MAX], proizvod[MAX][MAX];
    /* Dimenzija kvadratnih matrica */
79
    int n;
```

```
printf("Unesite broj vrsta matrica:\n");
     scanf("%d", &n);
83
     /* Proverava se da li je doslo do prekoracenja */
     if (n > MAX | | n \le 0) {
85
       fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
fprintf(stderr, "matrica.\n");
87
       exit(EXIT_FAILURE);
89
     printf("Unesite elemente prve matrice po vrstama:\n");
     ucitaj_matricu(a, n);
     printf("Unesite elemente druge matrice po vrstama:\n");
93
     ucitaj_matricu(b, n);
95
     /* Izracunava se zbir i proizvod matrica */
     saberi(a, b, zbir, n);
97
     pomnozi(a, b, proizvod, n);
99
     /* Ispisuje se rezultat */
     if (jednake_matrice(a, b, n) == 1)
       printf("Matrice su jednake.\n");
     else
       printf("Matrice nisu jednake.\n");
     printf("Zbir matrica je:\n");
     ispisi_matricu(zbir, n);
     printf("Proizvod matrica je:\n");
     ispisi_matricu(proizvod, n);
     exit(EXIT_SUCCESS);
113 }
```

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

#define MAX 64

/* Funkcija proverava da li je relacija refleksivna. Relacija je
    refleksivna ako je svaki element u relaciji sa sobom, odnosno ako
    se u matrici relacije na glavnoj dijagonali nalaze jedinice */
    int refleksivnost(int m[][MAX], int n)

{
    int i;

for (i = 0; i < n; i++) {
        if (m[i][i] != 1)</pre>
```

```
return 0;
    }
    return 1;
18
20
  /* Funkcija odredjuje refleksivno zatvorenje zadate relacije. Ono je
     odredjeno matricom koja sadrzi sve elemente polazne matrice
     dopunjene jedinicama na glavnoj dijagonali */
24 void ref_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
    int i, j;
26
    /* Prepisuju se vrednosti elemenata pocetne matrice */
28
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
30
        zatvorenje[i][j] = m[i][j];
    /* Na glavnoj dijagonali se postavljaju jedinice */
    for (i = 0; i < n; i++)
34
      zatvorenje[i][i] = 1;
  }
36
  /* Funkcija proverava da li je relacija simetricna. Relacija je
38
     simetricna ako za svaki par elemenata vazi: ako je element "i" u
     relaciji sa elementom "j", onda je i element "j" u relaciji sa
40
     elementom "i". Ovakve matrice su simetricne u odnosu na glavnu
     dijagonalu */
42
  int simetricnost(int m[][MAX], int n)
44
    int i, j;
46
    /* Obilaze se elementi ispod glavne dijagonale matrice i
       uporedjuju se sa njima simetricnim elementima */
48
    for (i = 0; i < n; i++)
      for (j = 0; j < i; j++)
        if (m[i][j] != m[j][i])
          return 0;
    return 1;
54
56
  /* Funkcija odredjuje simetricno zatvorenje zadate relacije. Ono je
     odredjeno matricom koja sadrzi sve elemente polazne matrice
     dopunjene tako da matrica postane simetricna u odnosu na glavnu
     dijagonalu */
60
  void sim_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
62
    int i, j;
64
    for (i = 0; i < n; i++)
     for (j = 0; j < n; j++)
66
```

```
zatvorenje[i][j] = m[i][j];
68
     for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
         if (zatvorenje[i][j] == 1)
           zatvorenje[j][i] = 1;
72
74
   /* Funkcija proverava da li je relacija tranzitivna. Relacija je
      tranzitivna ako ispunjava sledece svojstvo: ako je element "i" u
      relaciji sa elementom "j" i element "j" u relaciji sa elementom
      "k", onda je i element "i" u relaciji sa elementom "k" */
  int tranzitivnost(int m[][MAX], int n)
80
     int i, j, k;
82
     for (i = 0; i < n; i++)
84
       for (j = 0; j < n; j++)
         /* Ispituje se da li postoji element koji narusava *
86
            tranzitivnost */
         for (k = 0; k < n; k++)
88
           if (m[i][k] == 1 && m[k][j] == 1 && m[i][j] == 0)
             return 0:
90
     return 1;
92
94
   /* Funkcija odredjuje refleksivno-tranzitivno zatvorenje zadate
      relacije koriscenjem Varsalovog algoritma */
   void ref_tran_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
98
     int i, j, k;
     /* Prepisuju se vrednosti elemenata pocetne matrice */
     for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
104
         zatvorenje[i][j] = m[i][j];
106
     /* Odredjuje se reflektivno zatvorenje matrice */
     for (i = 0; i < n; i++)
108
       zatvorenje[i][i] = 1;
     /* Primenom Varsalovog algoritma odredjuje se tranzitivno
        zatvorenje matrice */
112
     for (k = 0; k < n; k++)
       for (i = 0; i < n; i++)
114
         for (j = 0; j < n; j++)
           if ((zatvorenje[i][k] == 1) && (zatvorenje[k][j] == 1)
               && (zatvorenje[i][j] == 0))
             zatvorenje[i][j] = 1;
```

```
120
   /* Funkcija ispisuje elemente matrice */
void pisi_matricu(int m[][MAX], int n)
     int i, j;
124
     for (i = 0; i < n; i++) {
126
       for (j = 0; j < n; j++)
         printf("%d ", m[i][j]);
128
       printf("\n");
130
   int main(int argc, char *argv[])
134 {
     FILE *ulaz;
     int m[MAX][MAX];
136
     int pomocna[MAX][MAX];
     int n, i, j;
138
     /* Provera da li korisnik nije uneo trazene argumente */
140
     if (argc < 2) {
       printf("Greska: ");
142
       printf("Nedovoljan broj argumenata komandne linije.\n");
       printf("Program se poziva sa %s ime_dat.\n", argv[0]);
       exit(EXIT_FAILURE);
146
     /* Otvara se datoteka za citanje */
     ulaz = fopen(argv[1], "r");
     if (ulaz == NULL) {
       fprintf(stderr, "Greska: ");
fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
       exit(EXIT_FAILURE);
154
     /* Ucitava se dimenzija matrice */
     fscanf(ulaz, "%d", &n);
158
     /* Proverava se da li je doslo do prekoracenja dimenzije */
     if (n > MAX || n \le 0) {
160
       fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
       fprintf(stderr, "matrice.\n");
       exit(EXIT_FAILURE);
164
     /* Ucitava se element po element matrice */
     for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
168
         fscanf(ulaz, "%d", &m[i][j]);
170
```

```
/* Ispisuje se rezultat */
     printf("Relacija %s refleksivna.\n",
            refleksivnost(m, n) == 1 ? "jeste" : "nije");
     printf("Relacija %s simetricna.\n",
            simetricnost(m, n) == 1 ? "jeste" : "nije");
     printf("Relacija %s tranzitivna.\n",
178
            tranzitivnost(m, n) == 1 ? "jeste" : "nije");
180
     printf("Refleksivno zatvorenje relacije:\n");
     ref_zatvorenje(m, n, pomocna);
182
     pisi_matricu(pomocna, n);
184
     printf("Simetricno zatvorenje relacije:\n");
     sim_zatvorenje(m, n, pomocna);
186
     pisi_matricu(pomocna, n);
188
     printf("Refleksivno-tranzitivno zatvorenje relacije:\n");
     ref_tran_zatvorenje(m, n, pomocna);
190
     pisi_matricu(pomocna, n);
192
     /* Zatvara se datoteka */
     fclose(ulaz);
194
     exit(EXIT_SUCCESS);
196
```

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

#define MAX 32

/* Funkcija izracunava najveci element na sporednoj dijagonali. Za
elemente sporedne dijagonale vazi da je zbir indeksa vrste i
indeksa kolone jednak n-1 */
int max_sporedna_dijagonala(int m[][MAX], int n)

{
   int i;
   int max_na_sporednoj_dijagonali = m[0][n - 1];

for (i = 1; i < n; i++)
   if (m[i][n - 1 - i] > max_na_sporednoj_dijagonali)
   max_na_sporednoj_dijagonali = m[i][n - 1 - i];

return max_na_sporednoj_dijagonali;

}

/* Funkcija izracunava indeks kolone najmanjeg elementa */
```

```
22 int indeks_min(int m[][MAX], int n)
  {
    int i, j;
24
    int min = m[0][0], indeks_kolone = 0;
26
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
28
        if (m[i][j] < min) {
          min = m[i][j];
30
          indeks_kolone = j;
    return indeks_kolone;
34
36
  /* Funkcija izracunava indeks vrste najveceg elementa */
38 int indeks_max(int m[][MAX], int n)
    int i, j;
40
    int max = m[0][0], indeks_vrste = 0;
42
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
44
        if (m[i][j] > max) {
          max = m[i][j];
46
          indeks_vrste = i;
        }
48
    return indeks_vrste;
50 }
52 /* Funkcija izracunava broj negativnih elemenata matrice */
  int broj_negativnih(int m[][MAX], int n)
54 {
    int i, j;
56
    int broj_negativnih = 0;
   for (i = 0; i < n; i++)
58
     for (j = 0; j < n; j++)
        if (m[i][j] < 0)
60
          broj_negativnih++;
    return broj_negativnih;
64 }
int main(int argc, char *argv[])
    int m[MAX][MAX];
68
    int n;
    int i, j;
70
    /* Proverava se broj argumenata komandne linije */
    if (argc < 2) {
```

```
printf("Greska: ");
74
       printf("Nedovoljan broj argumenata komandne linije.\n");
       printf("Program se poziva sa %s br_vrsta_mat.\n", argv[0]);
76
       exit(EXIT_FAILURE);
78
     /* Ucitava se broj vrsta matrice */
80
     n = atoi(argv[1]);
82
     if (n > MAX | | n <= 0) {
       fprintf(stderr, "Greska: Neodgovarajuci broj ");
fprintf(stderr, "vrsta matrice.\n");
84
       exit(EXIT_FAILURE);
86
88
     /* Ucitava se matrica */
     printf("Unesite elemente matrice dimenzije %dx%d:\n", n, n);
90
     for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
92
         scanf("%d", &m[i][j]);
94
     /* Ispisuju se rezultati izracunavanja */
     printf("Najveci element sporedne dijagonale je %d.\n",
96
            max_sporedna_dijagonala(m, n));
98
     printf("Indeks kolone sa najmanjim elementom je %d.\n",
             indeks_min(m, n));
     printf("Indeks vrste sa najvecim elementom je %d.\n",
             indeks_max(m, n));
104
     printf("Broj negativnih elemenata matrice je %d.\n",
            broj_negativnih(m, n));
106
     exit(EXIT_SUCCESS);
108
```

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

#define MAX 32

/* Funkcija ucitava elemente kvadratne matrice dimenzije n sa
standardnog ulaza */
void ucitaj_matricu(int m[][MAX], int n)
{
  int i, j;

for (i = 0; i < n; i++)</pre>
```

```
for (j = 0; j < n; j++)
        scanf("%d", &m[i][j]);
15 }
17 /* Funkcija ispisuje elemente kvadratne matrice dimenzije n na
     standardni izlaz */
void ispisi_matricu(int m[][MAX], int n)
    int i, j;
21
    for (i = 0; i < n; i++) {
     for (j = 0; j < n; j++)
        printf("%d ", m[i][j]);
      printf("\n");
  }
29
  /* Funkcija proverava da li je zadata matrica ortonormirana,
     odnosno, da li je normirana i ortogonalna. Matrica je normirana
     ako je proizvod svake vrste matrice sa samom sobom jednak
     jedinici. Matrica je ortogonalna, ako je proizvod dve bilo koje
     razlicite vrste matrice jednak nuli */
  int ortonormirana(int m[][MAX], int n)
35
    int i, j, k;
    int proizvod;
39
    /* Ispituje se uslov normiranosti */
    for (i = 0; i < n; i++) {
41
      proizvod = 0;
      for (j = 0; j < n; j++)
43
        proizvod += m[i][j] * m[i][j];
      if (proizvod != 1)
45
        return 0;
    }
47
    /* Ispituje se uslov ortogonalnosti */
49
    for (i = 0; i < n - 1; i++) {
      for (j = i + 1; j < n; j++) {
        proizvod = 0;
        for (k = 0; k < n; k++)
          proizvod += m[i][k] * m[j][k];
        if (proizvod != 0)
          return 0;
      }
    }
59
    /* Ako su oba uslova ispunjena, matrica je ortonormirana */
61
    return 1;
  int main()
```

```
65 {
    int A[MAX][MAX];
    int n;
67
    printf("Unesite broj vrsta matrice: ");
69
    scanf("%d", &n);
    if (n > MAX || n \le 0) {
      fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
      fprintf(stderr, "matrice.\n");
      exit(EXIT_FAILURE);
77
    printf("Unesite elemente matrice po vrstama:\n");
    ucitaj_matricu(A, n);
79
    printf("Matrica %s ortonormirana.\n",
81
           ortonormirana(A, n) ? "je" : "nije");
83
    exit(EXIT_SUCCESS);
  }
85
```

```
#include <stdio.h>
2 #include <stdlib.h>
 #define MAX 32
  /* Funkcija ucitava elemente kvadratne matrice dimenzije n sa
     standardnog ulaza */
  void ucitaj_matricu(int m[][MAX], int n)
    int i, j;
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
14
        scanf("%d", &m[i][j]);
16
  /* Funkcija ispisuje elemente kvadratne matrice dimenzije n na
    standardni izlaz */
  void ispisi_matricu(int m[][MAX], int n)
20 {
    int i, j;
22
    for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++)
        printf("%d ", m[i][j]);
      printf("\n");
26
```

```
28 }
  /* Funkcija proverava da li je zadata matrica ortonormirana,
30
     odnosno, da li je normirana i ortogonalna. Matrica je normirana
     ako je proizvod svake vrste matrice sa samom sobom jednak
     jedinici. Matrica je ortogonalna, ako je proizvod dve bilo koje
     razlicite vrste matrice jednak nuli */
34
  int ortonormirana(int m[][MAX], int n)
36 {
    int i, j, k;
    int proizvod;
38
    /* Ispituje se uslov normiranosti */
40
    for (i = 0; i < n; i++) {
      proizvod = 0;
42
      for (j = 0; j < n; j++)
        proizvod += m[i][j] * m[i][j];
44
      if (proizvod != 1)
        return 0;
46
    }
48
    /* Ispituje se uslov ortogonalnosti */
    for (i = 0; i < n - 1; i++) {
      for (j = i + 1; j < n; j++) {
        proizvod = 0;
        for (k = 0; k < n; k++)
          proizvod += m[i][k] * m[j][k];
        if (proizvod != 0)
          return 0;
56
      }
    /* Ako su oba uslova ispunjena, matrica je ortonormirana */
    return 1;
62 }
64 int main()
    int A[MAX][MAX];
    int n;
68
    printf("Unesite broj vrsta matrice: ");
    scanf("%d", &n);
70
    if (n > MAX || n \le 0) {
72
      fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
      fprintf(stderr, "matrice.\n");
74
      exit(EXIT_FAILURE);
    }
76
    printf("Unesite elemente matrice po vrstama:\n");
    ucitaj_matricu(A, n);
```

```
printf("Matrica %s ortonormirana.\n",
ortonormirana(A, n) ? "je" : "nije");

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
  #include <stdlib.h>
  int main()
5
  {
    int *p = NULL;
    int i, n;
    printf("Unesite dimenziju niza: ");
    scanf("%d", &n);
    /* Rezervise se prostor za n celih brojeva */
    if ((p = (int *) malloc(sizeof(int) * n)) == NULL) {
13
      fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
      exit(EXIT_FAILURE);
17
    printf("Unesite elemente niza: ");
    for (i = 0; i < n; i++)
19
      scanf("%d", &p[i]);
21
    printf("Niz u obrnutom poretku je: ");
    for (i = n - 1; i \ge 0; i--)
23
      printf("%d ", p[i]);
    printf("\n");
25
    /* Oslobadja se prostor rezervisan funkcijom malloc() */
27
    free(p);
29
    exit(EXIT_SUCCESS);
  }
31
```

```
#include <stdio.h>
2 #include <stdlib.h>
4 #define KORAK 10
6 int main()
```

```
/* Adresa prvog alociranog bajta */
    int *a = NULL:
    /* Velicina alocirane memorije */
    int alocirano;
    /* Broj elemenata niza */
14
    int n;
    /* Broj koji se ucitava sa ulaza */
18
    int x;
    int i:
    int *b = NULL;
20
    char realokacija;
    /* Inicijalizacija */
    alocirano = n = 0;
24
    printf("Unesite zeljeni nacin realokacije (M ili R):\n");
26
    scanf("%c", &realokacija);
28
    printf("Unesite brojeve, nulu za kraj:\n");
    scanf("%d", &x);
30
    while (x != 0) {
      if (n == alocirano) {
        alocirano = alocirano + KORAK;
34
        if (realokacija == 'M') {
36
          /* Vrsi se realokacija memorije sa novom velicinom */
          b = (int *) malloc(alocirano * sizeof(int));
38
          if (b == NULL) {
40
            fprintf(stderr, "Greska: Neuspesna alokacija memorije.\n");
            free(a);
42
            exit(EXIT_FAILURE);
          }
44
          /* Svih n elemenata koji pocinju na adresi a prepisuju se na
46
             novu aderesu b */
          for (i = 0; i < n; i++)
48
            b[i] = a[i];
50
          free(a);
          /* Promenljivoj a dodeljuje se adresa pocetka novog, veceg
             bloka cija je adresa prilikom alokacije zapamcena u
             promenljivoj b */
          a = b;
56
        } else if (realokacija == 'R') {
58
```

```
/* Zbog funkcije realloc je neophodno da i u prvoj iteraciji
              "a" bude inicijalizovano na NULL */
60
           a = (int *) realloc(a, alocirano * sizeof(int));
           if (a == NULL) {
62
             fprintf(stderr,
                     "Greska: Neupesna realokacija memorije.\n");
64
             exit(EXIT_FAILURE);
66
        }
      }
68
      a[n++] = x;
      scanf("%d", &x);
    printf("Niz u obrnutom poretku je: ");
72
    for (n--; n >= 0; n--)
      printf("%d ", a[n]);
74
    printf("\n");
76
    /* Oslobadja se dinamicki alocirana memorija */
    free(a);
78
    exit(EXIT_SUCCESS);
80
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #define MAX 1000
  /* Funkcija dinamicki kreira niz karaktera u koji smesta rezultat
     nadovezivanja niski. Adresa kreiranog niza se vraca kao povratna
     vrednost. */
  char *nadovezi(char *s, char *t)
    char *p = (char *) malloc((strlen(s) + strlen(t) + 1)
                               * sizeof(char));
14
    /* Proverava se da li je memorija uspesno alocirana */
    if (p == NULL) {
      fprintf(stderr, "Greska: Neuspesna alokacija memorije.\n");
      exit(EXIT_FAILURE);
18
20
    /* Kopiraju se i nadovezuju niske karaktera */
    strcpy(p, s);
    strcat(p, t);
24
    return p;
```

```
26 }
28 int main()
    char *s = NULL:
30
   char s1[MAX], s2[MAX];
32
    printf("Unesite dve niske karaktera:\n");
    scanf("%s", s1);
34
    scanf("%s", s2);
36
    /* Poziva se funkcija koja nadovezuje niske */
   s = nadovezi(s1, s2);
38
    /* Prikazuje se rezultat */
40
    printf("Nadovezane niske: %s\n", s);
42
    /* Oslobadja se memorija alocirana u funkciji nadovezi() */
    free(s);
44
    exit(EXIT_SUCCESS);
46
```

```
#include <stdio.h>
  #include <stdlib.h>
3 #include <math.h>
5 int main()
    int i, j;
    /* Pokazivac na niz vrsta matrice realnih brojeva */
9
    double **A = NULL;
    /* Broj vrsta i broj kolona */
    int n = 0, m = 0;
13
    /* Trag matice */
    double trag = 0;
17
    printf("Unesite broj vrsta i broj kolona:\n ");
19
    scanf("%d%d", &n, &m);
    /* Dinamicki se rezervise prostor za niz vrsta matrice */
    A = (double **) malloc(sizeof(double *) * n);
23
    /* Provera se da li je uspelo rezervisanje memorije */
    if (A == NULL) {
25
      fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
```

```
exit(EXIT_FAILURE);
29
    /* Dinamicki se rezervise prostor za elemente u vrstama */
    for (i = 0; i < n; i++) {
31
      A[i] = (double *) malloc(sizeof(double) * m);
33
      /* Ukoliko je alokacija neuspesna, pre zavrsetka programa
         potrebno je osloboditi svih i-1 prethodno alociranih vrsta, i
35
         alociran niz pokazivaca */
      if (A[i] == NULL) {
        for (j = 0; j < i; j++)
          free(A[j]);
39
        free(A);
        exit(EXIT_FAILURE);
41
43
    printf("Unesite elemente matrice po vrstama:\n");
45
    for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
47
        scanf("%lf", &A[i][j]);
49
    /* Izracunava se trag matrice, odnosno suma elemenata na glavnoj
       dijagonali */
    trag = 0.0;
53
    for (i = 0; i < n; i++)
      trag += A[i][i];
    printf("Trag unete matrice je %.2f.\n", trag);
    /* Oslobadja se prostor rezervisan za svaku vrstu */
59
    for (j = 0; j < n; j++)
61
      free(A[j]);
    /* Oslobadja se memorija za niz pokazivaca na vrste */
63
    free(A);
65
    exit(EXIT_SUCCESS);
  }
67
```

#### matrica.h

```
#ifndef _MATRICA_H_
#define _MATRICA_H_ 1

/* Funkcija dinamicki alocira memoriju za matricu dimenzije n x m */
```

```
5 int **alociraj_matricu(int n, int m);
7 /* Funkcija dinamicki alocira memoriju za kvadratnu matricu
    dimenzije n x n */
9 int **alociraj_kvadratnu_matricu(int n);
11 /* Funkcija oslobadja memoriju za matricu sa n vrsta */
  int **dealociraj_matricu(int **matrica, int n);
  /* Funkcija ucitava vec alociranu matricu dimenzije n x m sa
    standardnog ulaza */
  void ucitaj_matricu(int **matrica, int n, int m);
  /* Funkcija ucitava vec alociranu kvadratnu matricu dimenzije n sa
    standardnog ulaza */
19
  void ucitaj_kvadratnu_matricu(int **matrica, int n);
21
  /* Funkcija ispisuje matricu dimenzije n x m na standardnom izlazu */
void ispisi_matricu(int **matrica, int n, int m);
25 /* Funkcija ispisuje kvadratnu matricu dimenzije n na standardnom
    izlazu */
void ispisi_kvadratnu_matricu(int **matrica, int n);
29 /* Funkcija ucitava vec alociranu matricu dimenzije n x m iz
    datoteke f */
31 int ucitaj_matricu_iz_datoteke(int **matrica, int n, int m,
                                 FILE * f);
  /* Funkcija ucitava vec alociranu kvadratnu matricu dimenzije n x n
    iz datoteke f */
  int ucitaj_kvadratnu_matricu_iz_datoteke(int **matrica, int n,
                                           FILE * f);
39 /* Funkcija upisuje matricu dimenzije n x m u datoteku f */
  int upisi_matricu_u_datoteku(int **matrica, int n, int m, FILE * f);
  /* Funkcija upisuje kvadratnu matricu dimenzije n x n u datoteku f */
43 int upisi_kvadratnu_matricu_u_datoteku(int **matrica, int n,
                                         FILE * f):
45
  #endif
```

#### matrica.c

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

int **alociraj_matricu(int n, int m)
{
```

```
int **matrica = NULL;
    int i, j;
    /* Alocira se prostor za niz vrsta matrice */
    matrica = (int **) malloc(n * sizeof(int *));
    /* Ako alokacija nije prosla uspesno, povratna vrednost funkcije
       ce biti NULL, sto mora biti provereno u main funkciji */
    if (matrica == NULL)
14
      return NULL;
16
    /* Alocira se prostor za svaku vrstu matrice */
    for (i = 0; i < n; i++) {
18
      matrica[i] = (int *) malloc(m * sizeof(int));
      /* Ako alokacija nije prosla uspesno, oslobadjaju se svi
20
         prethodno alocirani resursi, i povratna vrednost je NULL */
      if (matrica[i] == NULL) {
        for (j = 0; j < i; j++)
          free(matrica[j]);
        free(matrica);
        return NULL;
26
28
    return matrica;
30
  int **alociraj_kvadratnu_matricu(int n)
    /* Alociranje matrice dimenzije n x n */
    return alociraj_matricu(n, n);
36
  int **dealociraj_matricu(int **matrica, int n)
    int i;
40
    /* Oslobadja se prostor rezervisan za svaku vrstu */
    for (i = 0; i < n; i++)
      free(matrica[i]);
    /* Oslobadja se memorija za niz pokazivaca na vrste */
    free(matrica);
46
    /* Matrica postaje prazna, tj. nealocirana */
    return NULL;
48
50
  void ucitaj_matricu(int **matrica, int n, int m)
52
    int i, j;
    /* Elementi matrice se ucitacaju po vrstama */
    for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
56
        scanf("%d", &matrica[i][j]);
58 }
```

```
60 void ucitaj_kvadratnu_matricu(int **matrica, int n)
     /* Ucitavanje matrice n x n */
     ucitaj_matricu(matrica, n, n);
64 }
66 void ispisi_matricu(int **matrica, int n, int m)
68
     int i, j;
    /* Ispis po vrstama */
    for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
        printf("%d ", matrica[i][j]);
72
       printf("\n");
74
   }
76
   void ispisi_kvadratnu_matricu(int **matrica, int n)
78 {
     /* Ispis matrice n x n */
    ispisi_matricu(matrica, n, n);
80
82
   int ucitaj_matricu_iz_datoteke(int **matrica, int n, int m, FILE * f)
84 {
    int i, j;
    /* Elementi matrice se ucitacaju po vrstama */
86
    for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
88
         /* Ako je nemoguce ucitati sledeci element, povratna vrednost
            funkcije je 1, kao indikator neuspesnog ucitavanja */
90
         if (fscanf(f, "%d", &matrica[i][j]) != 1)
           return 1;
92
     /* Uspesno ucitana matrica */
     return 0;
96 }
98 int ucitaj_kvadratnu_matricu_iz_datoteke(int **matrica, int n,
                                             FILE * f)
100 | {
     /* Ucitavanje matrice n x n iz datoteke */
     return ucitaj_matricu_iz_datoteke(matrica, n, n, f);
104
   int upisi_matricu_u_datoteku(int **matrica, int n, int m, FILE * f)
106 {
     int i, j;
    /* Ispis po vrstama */
108
    for (i = 0; i < n; i++) {
      for (j = 0; j < m; j++)
110
```

#### $main_a.c$

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include "matrica.h"
5 int main()
    int **matrica = NULL;
    int n, m;
    FILE *f;
    /* Ucitava se broj vrsta i broj kolona matrice */
    printf("Unesite broj vrsta matrice: ");
    scanf("%d", &n);
13
    printf("Unesite broj kolona matrice: ");
    scanf("%d", &m);
    /* Provera dimenzija matrice */
    if (n \le 0 \mid \mid m \le 0) {
      fprintf(stderr, "Greska: Broj vrsta i broj kolona ne mogu biti
19
      negativni brojevi.\n");
      exit(EXIT_FAILURE);
21
    }
    /* Rezervise se memorijski prostor za matricu i proverava se da li
       je memorijski prostor uspesno rezervisan */
    matrica = alociraj_matricu(n, m);
25
    if (matrica == NULL) {
      fprintf(stderr, "Greska: Neuspesna alokacija matrice.\n");
      exit(EXIT_FAILURE);
    }
29
```

```
/* Ucitava se matrica sa standardnog ulaza */
    printf("Unesite elemente matrice po vrstama:\n");
    ucitaj_matricu(matrica, n, m);
    /* Otvara se datoteka za upis matrice */
35
    if ((f = fopen("matrica.txt", "w")) == NULL) {
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke.\n");
      matrica = dealociraj_matricu(matrica, n);
      exit(EXIT_FAILURE);
    }
41
    /* Upis matrice u datoteku */
    if (upisi_matricu_u_datoteku(matrica, n, m, f) != 0) {
43
      fprintf(stderr, "Greska: Neuspesno upisivanje matrice u datoteku
      .\n");
      matrica = dealociraj_matricu(matrica, n);
45
      exit(EXIT_FAILURE);
47
    /* Zatvara se datoteka */
49
    fclose(f);
    /* Oslobadja se memorija koju je zauzimala matrica */
   matrica = dealociraj_matricu(matrica, n);
    exit(EXIT_SUCCESS);
```

#### $main_b.c$

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "matrica.h"
  int main(int argc, char **argv)
6 {
    int **matrica = NULL;
   int n;
    FILE *f;
    /* Provera argumenata komandne linije */
    if (argc != 2) {
12
      fprintf(stderr, "Greska: Koriscenje programa: %s datoteka\n",
      argv[0]);
      exit(EXIT_FAILURE);
14
    }
    /* Otvara se datoteka za citanje */
    if ((f = fopen(argv[1], "r")) == NULL) {
18
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke.\n");
20
      exit(EXIT_FAILURE);
```

```
/* Ucitava se dimenzija matrice */
    if (fscanf(f, "%d", &n) != 1) {
24
      fprintf(stderr, "Greska: Neispravan pocetak datoteke.\n");
      exit(EXIT_FAILURE);
26
28
    /* Provera se dimenzija matrice */
    if (n <= 0) {
30
      fprintf(stderr, "Greska: Neodgovarajca dimenzija matrice.\n");
      exit(EXIT_FAILURE);
34
    /* Rezervise se memorijski prostor za matricu i vrsi se provera */
    matrica = alociraj_kvadratnu_matricu(n);
36
    if (matrica == NULL) {
      fprintf(stderr, "Greska: Neuspesna alokacija matrice.\n");
38
      exit(EXIT_FAILURE);
40
    /* Ucitava se matrica iz datoteke */
42
    if (ucitaj_kvadratnu_matricu_iz_datoteke(matrica, n, f) != 0) {
      fprintf(stderr, "Greska: Neuspesno ucitavanje matrice iz datoteke
44
       .\n");
      matrica = dealociraj_matricu(matrica, n);
      exit(EXIT_FAILURE);
46
48
    /* Zatvara se datoteka */
    fclose(f);
    /* Ispis matrice na standardni izlaz */
    ispisi_kvadratnu_matricu(matrica, n);
54
    /* Oslobadja se memorija koju je zauzimala matrica */
    matrica = dealociraj_matricu(matrica, n);
56
    exit(EXIT_SUCCESS);
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "matrica.h"

/* Funkcija ispisuje elemente matrice ispod glavne dijagonale */
void ispisi_elemente_ispod_dijagonale(int **M, int n, int m)
{
```

```
int i, j;
    for (i = 0; i < n; i++) {
     for (j = 0; j <= i; j++)
        printf("%d ", M[i][j]);
13
      printf("\n");
  }
17
  int main()
19 {
    int m, n;
   int **matrica = NULL;
   printf("Unesite broj vrsta i broj kolona:\n ");
23
    scanf("%d %d", &n, &m);
    /* Rezervise se memorija za matricu */
    matrica = alociraj_matricu(n, m);
    /* Provera alokacije */
    if (matrica == NULL) {
     fprintf(stderr, "Greska: Neuspesna alokacija matrice.\n");
      exit(EXIT_FAILURE);
33
    printf("Unesite elemente matrice po vrstama:\n");
    ucitaj_matricu(matrica, n, m);
35
    printf("Elementi ispod glavne dijagonale matrice:\n");
    ispisi_elemente_ispod_dijagonale(matrica, n, m);
39
    /* Oslobadja se memorija */
    matrica = dealociraj_matricu(matrica, n);
41
    exit(EXIT_SUCCESS);
43
```

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

/* Funkcija izvrsava trazene transformacije nad matricom */
void izmeni(float **a, int n)
{
   int i, j;

for (i = 0; i < n; i++)
   for (j = 0; j < n; j++)
   if (i < j)</pre>
```

```
a[i][j] /= 2;
         else if (i > j)
14
           a[i][j] *= 2;
  }
16
  /* Funkcija izracunava zbir apsolutnih vrednosti elemenata ispod
     sporedne dijagonale. Element se nalazi ispod sporedne dijagonale
     ukoliko je zbir indeksa vrste i indeksa kolone elementa veci od
20
22 float zbir_ispod_sporedne_dijagonale(float **m, int n)
    int i, j;
24
    float zbir = 0;
26
    for (i = 0; i < n; i++)
      for (j = n - i; j < n; j++)
2.8
        if (i + j > n - 1)
          zbir += fabs(m[i][j]);
30
    return zbir;
32
34
  /* Funkcija ucitava elemente kvadratne matrice dimenzije n x n iz
     zadate datoteke */
36
  void ucitaj_matricu(FILE * ulaz, float **m, int n)
  {
38
    int i, j;
40
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
42
        fscanf(ulaz, "%f", &m[i][j]);
  }
44
  /* Funkcija ispisuje elemente kvadratne matrice dimenzije n x n na
     standardni izlaz */
  void ispisi_matricu(float **m, int n)
48
    int i, j;
    for (i = 0; i < n; i++) {
      for (j = 0; j < n; j++)
        printf("%.2f ", m[i][j]);
54
      printf("\n");
    }
56
58
  /* Funkcija alocira memoriju za kvadratnu matricu dimenzije n x n */
60 float **alociraj_memoriju(int n)
    int i, j;
    float **m;
64
```

```
m = (float **) malloc(n * sizeof(float *));
     if (m == NULL) {
       fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
       exit(EXIT_FAILURE);
     for (i = 0; i < n; i++) {
      m[i] = (float *) malloc(n * sizeof(float));
       if (m[i] == NULL) {
74
         fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
         for (j = 0; j < i; j++)
          free(m[i]);
         free(m);
         exit(EXIT_FAILURE);
80
     }
     return m;
82
84
   /* Funckija oslobadja memoriju zauzetu kvadratnom matricom dimenzije
     n x n */
86
   void oslobodi_memoriju(float **m, int n)
88 | {
     int i;
90
     for (i = 0; i < n; i++)
      free(m[i]);
92
     free(m);
94 }
96 int main(int argc, char *argv[])
    FILE *ulaz;
98
    float **a;
     int n;
     /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
        greska */
     if (argc < 2) {
104
      printf("Greska: ");
       printf("Nedovoljan broj argumenata komandne linije.\n");
106
       printf("Program se poziva sa %s ime_dat.\n", argv[0]);
       exit(EXIT_FAILURE);
108
     /* Otvara se datoteka za citanje */
     ulaz = fopen(argv[1], "r");
112
     if (ulaz == NULL) {
      fprintf(stderr, "Greska: ");
114
       fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
116
       exit(EXIT_FAILURE);
```

```
/* Cita se dimenzija matrice */
     fscanf(ulaz, "%d", &n);
     /* Rezervise se memorija */
     a = alociraj_memoriju(n);
124
     /* Ucitavaju se elementi matrice */
     ucitaj_matricu(ulaz, a, n);
126
     float zbir = zbir_ispod_sporedne_dijagonale(a, n);
128
     /* Poziva se funkcija za transformaciju matrice */
130
     izmeni(a, n);
     /* Ispisuju se rezultati */
     printf("Zbir apsolutnih vrednosti ispod sporedne dijagonale ");
134
     printf("je %.2f.\n", zbir);
136
     printf("Transformisana matrica je:\n");
     ispisi_matricu(a, n);
138
     /* Oslobadja se memorija */
140
     oslobodi_memoriju(a, n);
142
     /* Zatvara se datoteka */
     fclose(ulaz);
144
     exit(EXIT_SUCCESS);
146
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <math.h>
  #include <string.h>
  /* Funkcija tabela() prihvata granice intervala a i b, broj
     ekvidistantnih tacaka n, kao i pokazivac f koji pokazuje na
     funkciju koja prihvata double argument, i vraca double vrednost.
     Za tako datu funkciju ispisuju se njene vrednosti u intervalu
     [a,b] u n ekvidistantnih tacaka intervala */
  void tabela(double a, double b, int n, double (*fp) (double))
  {
13
    int i;
    double x;
    printf("----\n");
    for (i = 0; i < n; i++) {
```

```
x = a + i * (b - a) / (n - 1);
      printf("| %8.5f | %8.5f |\n", x, (*fp) (x));
19
    printf("----
23
  double sqr(double a)
25 | {
    return a * a;
27 }
29 int main(int argc, char *argv[])
    double a, b;
    int n;
    char ime_funkcije[6];
    /* Pokazivac na funkciju koja ima jedan argument tipa double i
       povratnu vrednost istog tipa */
    double (*fp) (double);
39
    /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
       greska */
41
    if (argc < 2) {
      fprintf(stderr, "Greska: ");
43
      fprintf(stderr, "Nedovoljan broj argumenata komandne linije.\n");
      fprintf(stderr,
45
              "Program se poziva sa %s ime_funkcije iz math.h.\n",
              argv[0]);
47
      exit(EXIT_FAILURE);
49
    /* Niska ime_funkcije sadrzi ime trazene funkcije koja je navedena
       u komandnoj liniji */
    strcpy(ime_funkcije, argv[1]);
    /* Inicijalizuje se pokazivac na funkciju koja se tabelira */
    if (strcmp(ime_funkcije, "sin") == 0)
     fp = &sin;
    else if (strcmp(ime_funkcije, "cos") == 0)
      fp = &cos;
59
    else if (strcmp(ime_funkcije, "tan") == 0)
      fp = &tan;
61
    else if (strcmp(ime_funkcije, "atan") == 0)
      fp = &atan;
    else if (strcmp(ime_funkcije, "acos") == 0)
      fp = &acos;
65
    else if (strcmp(ime_funkcije, "asin") == 0)
     fp = &asin;
67
    else if (strcmp(ime_funkcije, "exp") == 0)
69
      fp = &exp;
```

```
else if (strcmp(ime_funkcije, "log") == 0)
      fp = &log;
    else if (strcmp(ime_funkcije, "log10") == 0)
      fp = &log10;
    else if (strcmp(ime_funkcije, "sqrt") == 0)
      fp = &sqrt;
    else if (strcmp(ime_funkcije, "floor") == 0)
      fp = &floor;
    else if (strcmp(ime_funkcije, "ceil") == 0)
      fp = &ceil;
79
    else if (strcmp(ime_funkcije, "sqr") == 0)
      fp = &sqr;
81
    else {
      fprintf(stderr, "Greska");
83
      fprintf(stderr,
               "Program jos uvek ne podrzava trazenu funkciju!\n");
85
      exit(EXIT_FAILURE);
87
    printf("Unesite krajeve intervala:\n");
89
    scanf("%lf %lf", &a, &b);
91
    printf("Koliko tacaka ima na ekvidistantnoj mrezi ");
    printf("(ukljucujuci krajeve intervala)?\n");
93
    scanf("%d", &n);
95
    /* Mreza mora da ukljucuje bar krajeve intervala, tako da se mora
       uneti broj veci od 2 */
97
    if (n < 2) {
      fprintf(stderr, "Greska: Broj tacaka mreze mora biti bar 2!\n");
99
      exit(EXIT_FAILURE);
    /* Ispisuje se ime funkcije */
                x %10s(x)\n", ime_funkcije);
    printf("
    /* Prosledjuje se funkciji tabela() funkcija zadata kao argument
       komandne linije */
    tabela(a, b, n, fp);
    exit(EXIT_SUCCESS);
  }
```

# Algoritmi pretrage i sortiranja

# 3.1 Algoritmi pretrage

**Zadatak 3.1** Napisati iterativne funkcije za pretragu nizova. Svaka funkcija treba da vrati indeks pozicije na kojoj je pronađen traženi broj ili vrednost -1 ukoliko broj nije pronađen.

- (a) Napisati funkciju linarna\_pretraga koja vrši linearnu pretragu niza celih brojeva a, dužine n, tražeći u njemu broj x.
- (b) Napisati funkciju binarna\_pretraga koja vrši binarnu pretragu sortiranog niza a, dužine n, tražeći u njemu broj x.
- (c) Napisati funkciju interpolaciona\_pretraga koja vrši interpolacionu pretragu sortiranog niza a, dužine n, tražeći u njemu broj x.

Napisati i program koji generiše rastući niz slučajnih brojeva dimenzije n i pozivajući napisane funkcije traži broj x. Programu se kao prvi argument komandne linije prosleđuje prirodan broj  $\mathbf n$  koji nije veći od 1000000 i broj  $\mathbf n$  kao drugi argument komandne linije. Potrebna vremena za izvršavanje ovih funkcija dopisati u datoteku vremena.txt.

#### Test 1

```
POKRETANJE: ./a.out 1000000 23542
                                                     VREMENA.TXT
                                                       Dimenzija niza: 1000000
IZLAZ:
                                                       Linearna: 3615091 ns
  Linearna pretraga:
                                                       Binarna: 1536 ns
 Element nije u nizu
                                                        Interpolaciona: 558 ns
 Binarna pretraga:
 Element nije u nizu
 Interpolaciona pretraga:
 Element nije u nizu
  Test 2
                                                     VREMENA.TXT
POKRETANJE: ./a.out 100000 37842
                                                      Dimenzija niza: 1000000
                                                       Linearna: 3615091 ns
 Linearna pretraga:
                                                        Binarna: 1536 ns
 Element nije u nizu
                                                       Interpolaciona: 558 ns
 Binarna pretraga:
 Element nije u nizu
                                                       Dimenzija niza: 100000
  Interpolaciona pretraga:
                                                       Linearna: 360803 ns
 Element nije u nizu
```

Zadatak 3.2 Napisati rekurzivne funkcije algoritama linearne, binarne i interpolacione pretrage i program koji ih testira za brojeve koji se unose sa standardnog ulaza. Linearnu pretragu implementirati na dva načina, svođenjem pretrage na prefiks i na sufiks niza. Prvo se unosi broj koji se traži, a zatim sortirani elementi niza sve do kraja ulaza. Pretpostaviti da niz brojeva koji se unosi neće biti duži od 1024 elemenata.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite trazeni broj: 11
Unesite sortiran niz elemenata:
2 5 6 8 10 11 23
Linearna pretraga
Pozicija elementa je 5.
Binarna pretraga
Pozicija elementa je 5.
Interpolaciona pretraga
Pozicija elementa je 5.
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite trazeni broj: 14
Unesite sortiran niz elemenata:
10 32 35 43 66 89 100
Linearna pretraga
Element se ne nalazi u nizu.
Binarna pretraga
Element se ne nalazi u nizu.
Interpolaciona pretraga
Element se ne nalazi u nizu.
```

Binarna: 1187 ns Interpolaciona: 628 ns

Zadatak 3.3 Napisati program koji preko argumenta komandne linije dobija ime datoteke koja sadrži sortirani spisak studenta po broju indeksa rastuće. Za svakog studenta u jednom redu stoje informacije o indeksu, imenu i prezimenu. Program učitava spisak studenata u niz i traži od korisnika indeks ili prezime studenta čije informacije se potom prikazuju na ekranu. U slučaju više

studenata sa istim prezimenom prikazati informacije o prvom takvom. Odabir kriterijuma pretrage se vrši kroz poslednji argument komandne linije, koji može biti -indeks ili -prezime. U slučaju neuspešnih pretragi, štampati odgovarajuću poruku. Pretrage implementirati u vidu iterativnih funkcija što manje složenosti. Pretpostaviti da u datoteci neće biti više od 128 studenata i da su imena i prezimena svih kraća od 16 slova.

#### Primer 1

20140104 Ivan Popovic 20140187 Vlada Stankovic 20140234 Darko Brankovic

```
POKRETANJE: ./a.out datoteka.txt -indeks
                                                  INTERAKCIJA SA PROGRAMOM:
                                                    Unesite indeks studenta
                                                    cije informacije zelite:
DATOTEKA.TXT
                                                    20140076
 20140003 Marina Petrovic
 20140012 Stefan Mitrovic
                                                    Indeks: 20140076.
 20140032 Dejan Popovic
                                                    Ime i prezime: Sonja Stevanovic
 20140049 Mirko Brankovic
 20140076 Sonja Stevanovic
 20140104 Ivan Popovic
 20140187 Vlada Stankovic
 20140234 Darko Brankovic
 Primer 2
POKRETANJE: ./a.out datoteka.txt -prezime
                                                  INTERAKCIJA SA PROGRAMOM:
                                                    Unesite prezime studenta
                                                    cije informacije zelite:
DATOTEKA.TXT
 20140003 Marina Petrovic
                                                    Popovic
                                                    Indeks: 20140032,
 20140012 Stefan Mitrovic
                                                    Ime i prezime: Dejan Popovic
 20140032 Dejan Popovic
 20140049 Mirko Brankovic
 20140076 Sonja Stevanovic
```

Zadatak 3.4 Modifikovati zadatak 3.3 tako da tražene funkcije budu rekurzivne.

**Zadatak 3.5** U datoteci koja se zadaje kao prvi argument komandne linije, nalaze se koordinate tačaka. U zavisnosti od prisustva opcija komandne linije (-x ili -y), pronaći onu koja je najbliža x, ili y osi, ili koordinatnom početku, ako nije prisutna nijedna opcija. Pretpostaviti da je broj tačaka u datateci veći od 0 i ne veći od 1024.

```
Test 1
                                                     Test 2
POKRETANJE: ./a.out dat.txt -x
                                                  POKRETANJE: ./a.out dat.txt
DAT.TXT
                                                   DAT.TXT
                                                     12 53
 12 53
 2.342 34.1
                                                     2.342 34.1
 -0.3 23
                                                     -0.3 23
 -1 23.1
                                                     -1 2.1
 123.5 756.12
                                                     123.5 756.12
IZLAZ:
                                                   IZLAZ:
 -0.3 23
```

Zadatak 3.6 Napisati funkciju koja određuje nulu funkcije cos(x) na intervalu [0,2] metodom polovljenja intervala. Algoritam se završava kada se vrednost kosinusne funkcije razlikuje za najviše 0.001 od nule. UPUTSTVO: Korisiti metod polovljenja intervala (algoritam analogan algoritmu binarne pretrage). NAPOMENA: Ovaj metod se može primeniti na funkciju cos(x) na intervalu [0,2] zato što je ona na ovom intervalu neprekidna, i vrednosti funkcije na krajevima intervala su različitog znaka.

```
Test 1
```

Zadatak 3.7 Napisati funkciju koja metodom polovljenja intervala određuje nulu izabrane funkcije na proizvoljnom intervalu sa tačnošću *epsilon*. Ime funkcije se zadaje kao prvi agrument komandne linije, a interval i tačnost se unose sa standardnog ulaza. Pretpostaviti da je izabrana funkcija na tom intervalu neprekidna. UPUTSTVO: *U okviru algoritma pretrage koristiti pokazivač na odgovarajuću funkciju (na primer, kao u zadatku 2.27).* 

```
Primer 1
                                                  Primer 2
POKRETANJE: ./a.out cos
                                                 POKRETANJE: ./a.out sin
INTERAKCIJA SA PROGRAMOM:
                                                 INTERAKCIJA SA PROGRAMOM:
 Unesite krajeve intervala:
                                                  Unesite krajeve intervala:
 Unesite preciznost: 0.001
                                                  Unesite preciznost: 0.00001
 1.57031
 Primer 3
                                                  Primer 4
POKRETANJE: ./a.out tan
                                                 POKRETANJE: ./a.out sin
INTERAKCIJA SA PROGRAMOM:
                                                 INTERAKCIJA SA PROGRAMOM:
 Unesite krajeve intervala: -1.1 1
                                                  Unesite krajeve intervala: 1 3
 Unesite preciznost: 0.00001
                                                  Funkcija sin na intervalu [1, 3]
 3.8147e-06
                                                  ne zadovoljava uslove
```

Zadatak 3.8 Napisati funkciju koja u rastuće sortiranom nizu celih brojeva binarnom pretragom pronalazi indeks prvog elementa većeg od nule. Ukoliko nema elemenata većih od nule, funkcija kao rezultat vraća -1. Napisati program koji testira ovu funkciju za rastući niz celih brojeva koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

**Zadatak 3.9** Napisati funkciju koja u opadajuće sortiranom nizu celih brojeva binarnom pretragom pronalazi indeks prvog elementa manjeg od nule. Ukoliko nema elemenata manjih od nule, funkcija kao rezultat vraća -1. Napisati program koji testira ovu funkciju za opadajući niz celih brojeva koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

Zadatak 3.10 Napisati funkciju koja određuje ceo deo logaritma za osnovu 2 datog neoznačenog celog broja koristeći samo bitske i relacione operatore.

- (a) Napisati funkciju linearne složenosti koja određuje logaritam pomeranjem broja udesno.
- (b) Napisati funkciju logaritmske složenosti koja određuje logaritam koristeći binarnu pretragu.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati pozitivan ceo broj a na standardni izlaz ispisati njegov logaritam.

Test 1	Test 2	Test 3
ULAZ:	ULAZ:   17	ULAZ: 1031
IZLAZ: 2 2	IZLAZ: 4 4	IZLAZ: 10 10

\* Zadatak 3.11 U prvom kvadrantu dato je  $1 \leq N \leq 10000$  duži svojim koordinatama (duži mogu da se seku, preklapaju, itd.). Napisati program koji pronalazi najmanji ugao  $0 \leq \alpha \leq 90^\circ$ , na dve decimale, takav da je suma dužina duži sa obe strane polupoluprave iz koordinatnog početka pod uglom  $\alpha$  jednak (neke duži bivaju presečene, a neke ne). Program prvo učitava broj N, a zatim i same koordinate temena duži. UPUTSTVO:  $Vršiti\ binarnu\ pretragu\ intervala\ [0,90°].$ 

Primer 1	Primer 2	Primer 3
Interakcija sa programom: Unesite broj tacaka: 2 Unesite koordinate tacaka: 2 0 2 1 1 2 2 2 26.57	INTERAKCIJA SA PROGRAMOM: Unesite broj tacaka: 2 Unesite koordinate tacaka: 1 0 1 1 0 1 1 1 45	INTERAKCIJA SA PROGRAMOM: Unesite broj tacaka: 3 Unesite koordinate tacaka: 1 0 1 1 2 0 2 1 1 2 2 2 26.57

# 3.2 Algoritmi sortiranja

Zadatak 3.12 Napraviti biblioteku koja implementira algoritme sortiranja nizova celih brojeva. Biblioteka treba da sadrži algoritam sortiranja izborom (engl. selection sort), sortiranja spajanjem (engl. merge sort), brzog sortiranja (engl. quick sort), mehurastog sortiranja (engl. bubble sort), sortiranja direktnim umetanjem (engl. insertion sort) i sortiranja umetanjem sa inkrementom (engl. shell sort). Upotrebiti biblioteku kako bi se napravilo poređenje efikasnosti različitih algoritama sortiranja. Efikasnost meriti na slučajno generisanim nizovima, na rastuće sortiranim nizovima i na opadajuće sortiranim nizovima. Izbor algoritma, veličine i početnog rasporeda elemenata niza birati kroz argumente komandne linije. Moguće opcije kojima se bira algoritam sortiranja su: -m za sortiranje spajanjem, -q za brzo sortiranje, -b za mehurasto, -i za sortiranje direktnim umetanjem ili -s za sortiranje umetanjem sa inkrementom. U slučaju da nije prisutna ni jedna od ovih opcija, niz sortirati algoritmom sortiranja izborom. Niz koji se sortira generisati neopadajuće ako je prisutna opcija

-r, nerastuće ako je prisutna opcija -o ili potpuno slučajno ako nema nijedne opcije. Vreme meriti programom time. Analizirati porast vremena sa porastom dimenzije n.

```
Test 1
                                 Test 2
                                                                Test 3
                                                              || POKRETANJE: time ./a.out
|| POKRETANJE: time ./a.out
                               || POKRETANJE: time ./a.out
       200000
                                      400000
                                                                      800000
 TZLAZ:
                                 TZLAZ:
                                                                TZI.AZ:
  real 0m42.168s
                                  real 2m48.395s
                                                                 real 11m13.703s
   user 0m42.100s
                                  user 2m48.128s
                                                                  user 11m12.636s
   sys 0m0.000s
                                  sys 0m0.000s
                                                                  sys 0m0.000s
                                 Test 5
                                                                Test 6
  Test 4
|| POKRETANJE: time ./a.out
                               | POKRETANJE: time ./a.out
                                                              POKRETANJE: time ./a.out
                                      p- 000008
       800000 -r
                                                                      800000 -m
 IZLAZ:
                                 IZLAZ:
                                                                IZLAZ:
   real 11m21.533s
                                  real 0m0.159s
                                                                  real 0m0.137s
   user 11m20.436s
                                  user 0m0.156s
                                                                  user 0m0.136s
   sys 0m0.020s
                                  sys 0m0.000s
                                                                  sys 0m0.000s
```

**Zadatak 3.13** Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Napisati program koji proverava da li su dve niske karaktera anagrami. Niske se zadaju sa standardnog ulaza i neće biti duže od 127 karaktera. UPUTSTVO: Napisati funkciju koja sortira slova unutar niske karaktera, a zatim za sortirane niske proveriti da li su identične.

Primer 1	Primer 2	Primer 3
INTERAKCIJA SA PROGRAMOM: Unesite prvu nisku anagram Unesite drugu nisku ramgana jesu	Unesite prvu nisku anagram	INTERAKCIJA SA PROGRAMOM: Unesite prvu nisku test Unesite drugu nisku tset jesu

Zadatak 3.14 U datom nizu brojeva treba pronaći dva broja koja su na najmanjem rastojanju. Niz se zadaje sa standardnog ulaza, sve do kraja ulaza, ali neće sadržati više od 256 i manje od 2 elemenata. Na izlaz ispisati razliku pronađena dva broja. UPUTSTVO: Prvo sortirati niz. NAPOMENA: Koristiti biblioteku za sortiranje celih brojeva iz zadatka 3.12.

Zadatak 3.15 Napisati program koji pronalazi broj koji se najviše puta pojavljivao u datom nizu. Niz se zadaje sa standardnog ulaza sve do kraja ulaza i neće biti duži od 256 i kraći od jednog elemenata. UPUTSTVO: Prvo sortirati niz, a zatim naći najdužu sekvencu jednakih elemenata. NAPOMENA: Koristiti biblioteku za sortiranje celih brojeva iz zadatka 3.12.

Test 1	Test 2	Test 3
ULAZ: 4 23 5 2 4 6 7 34 6 4 5	ULAZ: 2 4 6 2 6 7 99 1	ULAZ: 123
IZLAZ:	Izlaz:	IzLAZ:

Zadatak 3.16 Napisati funkciju koja proverava da li u datom nizu postoje dva elementa čiji zbir je jednak zadatom celom broju. Napisati i program koji testira ovu funkciju. U programu se prvo učitava broj, a zatim i niz. Elementi niza se unose sve do kraja ulaza. Pretpostaviti da u niz neće biti uneto više od 256 brojeva. UPUTSTVO: Prvo sortirati niz. NAPOMENA: Koristiti biblioteku za sortiranje celih brojeva iz zadatka 3.12.

Primer 1	Primer 2	Primer 3
Interakcija sa programom: Unesite trazeni zbir: 34 Unesite elemente niza: 134 4 1 6 30 23 da	INTERAKCIJA SA PROGRAMOM: Unesite trazeni zbir: 12 Unesite elemente niza: 53 1 43 3 56 13 ne	INTERAKCIJA SA PROGRAMOM: Unesite trazeni zbir: 52 Unesite elemente niza: 52 ne

Zadatak 3.17 Napisati funkciju potpisa int merge(int \*niz1, int dim1, int \*niz2, int dim2, int \*niz3, int dim3) koja prima dva sortirana niza, i na osnovu njih pravi novi sortirani niz koji sadrži elemente oba niza. Treća dimenzija predstavlja veličinu niza u koji se smešta rezultat. Ako je ona manja od potrebne dužine, funkcija vraća -1 kao indikator neuspeha, inače vraća 0. Napisati zatim program koji testira ovu funkciju. Nizovi se unose sa standardnog ulaza sve dok se ne unese 0 i može se pretpostaviti da će njihove dimenzije biti

manje od 256.

```
        Primer 1
        Primer 2

        Interakcija sa programom:
        Unesite elemente prvog niza:
        Unesite elemente prvog niza:
        Unesite elemente prvog niza:
        1 4 7 0

        Unesite elemente drugog niza:
        Unesite elemente drugog niza:
        9 11 23 54 75 0
        9 11 23 54 75 0

        3 3 5 6 7 8 11 14 35
        1 4 7 9 11 23 54 75
        1 2 7 9 11 23 54 75
```

Zadatak 3.18 Napisati program koji čita sadržaj dve datoteke od kojih svaka sadrži spisak imena i prezimena studenata iz jedne od dve grupe, rastuće sortiran po imenima, a u slučaju istog imena po prezimenima, i kreira jedinstven spisak studenata sortiranih takođe po istom kriterijumu. Program dobija nazive datoteka iz komandne linije i jedinstveni spisak upisuje u datoteku ceo-tok.txt. Pretpostaviti da ime studenta nije duže od 10, a prezime od 15 karaktera.

Test 1

```
POKRETANJE: ./a.out prvi-deo.txt drugi-deo.txt
PRVI-DEO.TXT
                                                   CEO-TOK.TXT
                                                     Aleksandra Cvetic
 Andrija Petrovic
                                                     Andrija Petrovic
 Anja Ilic
                                                     Anja Ilic
 Ivana Markovic
                                                     Bojan Golubovic
 Lazar Micic
 Nenad Brankovic
                                                    Dragan Markovic
                                                    Filip Dukic
 Sofija Filipovic
                                                     Ivana Markovic
 Uros Milic
 Vladimir Savic
                                                     Ivana Stankovic
                                                     Lazar Micic
                                                     Marija Stankovic
DRUGI-DEO.TXT
                                                     Nenad Brankovic
 Aleksandra Cvetic
                                                     Ognjen Peric
 Bojan Golubovic
                                                     Sofija Filipovic
 Dragan Markovic
 Filip Dukic
                                                     Vladimir Savic
                                                     Uros Milic
 Ivana Stankovic
 Marija Stankovic
 Ognjen Peric
```

Zadatak 3.19 Napisati funkcije koje sortiraju niz struktura tačaka na osnovu sledećih kriterijuma: (i) njihovog rastojanja od koordinatnog početka, (ii) x koordinata tačaka, (iii) y koordinata tačaka. Napisati program koji učitava niz tačaka iz datoteke čije se ime zadaje kao drugi argument komandne linije, i u zavisnosti od prisutnih opcija (prvi argument) u komandnoj liniji (-o, -x ili -y) sortira tačke po jednom od prethodna tri kriterijuma i rezultat upisuje u datoteku čije se ime zadaje kao treći argument komandne linije. U ulaznoj datoteci nije zadato više od 128 tačaka.

```
Test 1
                                                    Test 2
POKRETANJE: ./a.out -x in.txt out.txt
                                                  POKRETANJE: ./a.out -o in.txt out.txt
IN.TXT
                                                  IN.TXT
 3 4
                                                    3 4
 11 6
                                                    11 6
 7 3
                                                    7 3
 2 82
                                                    2 82
 -1 6
                                                    -1 6
OUT.TXT
                                                  OUT.TXT
 -1 6
                                                   3 4
 2 82
                                                    -1 6
 3 4
                                                    7 3
 7 3
                                                    11 6
 11 6
                                                    2 82
```

Zadatak 3.20 Napisati program koji učitava imena i prezimena građana (najviše njih 1000) iz datoteke biracki-spisak.txt i kreira biračke spiskove. Jedan birački spisak je sortiran po imenu građana, a drugi po prezimenu. Program treba da ispisuje koliko građana ima isti redni broj u oba biračka spiska. Pretpostaviti da je za ime, odnosno prezime građana dovoljno 15 karaktera, i da se nijedno ime i prezime ne pojavljuje više od jednom.

```
Test 1
                               Test 2
                                                              Test 3
BIRACKI-SPISAK.TXT
                             | BIRACKI-SPISAK.TXT
                                                             | DATOTEKA BIRACKI-SPISAK.TXT
 Bojan Golubovic
                                Milan Milicevic
                                                              NE POSTOJI
 Andrija Petrovic
 Anja Ilic
                              IZLAZ:
                                                              IZLAZ ZA GREŠKE:
 Aleksandra Cvetic
                               1
                                                               Neupesno otvaranje
 Dragan Markovic
                                                              datoteke za citanje
 Ivana Markovic
 Lazar Micic
 Marija Stankovic
 Filip Dukic
IZLAZ:
 3
```

Zadatak 3.21 Definisati strukturu koja čuva imena, prezimena i godišta dece. Pretpostaviti da su imena i prezimena niske karaktera koje nisu duže od 30 karaktera. Napisati funkciju koja sortira niz dece po godištu, a decu istog godišta sortira leksikografski po prezimenu i imenu. Napisati program koji učitava podatke o deci koji se nalaze u datoteci čije se ime zadaje kao prvi argument komandne linije, sortira ih i sortirani niz upisuje u datoteku čije se ime zadaje kao drugi argument komandne linije. Pretpostaviti da u ulaznoj datoteci nisu zadati podaci o više od 128 dece.

#### Test 1

```
POKRETANJE: ./a.out in.txt out.txt
                                                  OUT.TXT
IN.OUT
 Petar Petrovic 2007
                                                   Marija Antic 2007
 Milica Antonic 2008
                                                    Ana Petrovic 2007
                                                    Petar Petrovic 2007
 Ana Petrovic 2007
                                                   Milica Antonic 2008
 Ivana Ivanovic 2009
 Dragana Markovic 2010
                                                    Ivana Ivanovic 2009
                                                    Dragana Markovic 2010
 Marija Antic 2007
  Test 2
POKRETANJE: ./a.out in.txt out.txt
                                                  OUT.TXT
TN.OUT
 Milijana Maric 2009
                                                   Milijana Maric 2009
```

Zadatak 3.22 Napisati funkciju koja sortira niz niski po broju suglasnika u niski. Ukoliko reči imaju isti broj suglasnika, sortirati ih po dužini niske rastuće, a ukoliko su i dužine jednake onda leksikografski rastuće. Napisati program koji testira ovu funkciju za niske koje se zadaju u datoteci niske.txt. Pretpostaviti da u nizu nema više od 128 elemenata, kao i da svaka niska sadrži najviše 31 karakter.

```
Test 1

NISKE.TXT
ana petar andjela milos nikola aleksandar ljubica matej milica

IZLAZ:
ana matej milos petar milica nikola andjela ljubica aleksandar
```

Zadatak 3.23 Napisati program koji simulira rad kase u prodavnici. Kupci prilaze kasi, a prodavac unošenjem bar-koda kupljenog proizvoda dodaje njegovu cenu na ukupan račun. Na kraju, program ispisuje ukupnu vrednost svih proizvoda. Sve artikle, zajedno sa bar-kodovima, prozivođačima i cenama učitati iz datoteke artikli.txt. Pretraživanje niza artikala vršiti binarnom pretragom.

## ARTIKLI, TXT 1001 Keks Jaffa 120

Primer 1

```
2530 Napolitanke Bambi 230
 0023 MedenoSrce Pionir 150
 2145 Pardon Marbo 70
INTERAKCIJA SA PROGRAMOM:
 KOD Naziv artikla Ime proizvodjaca Cena
  23 MedenoSrce Pionir 150.00
  1001 Keks Jaffa 120.00
  2145 Pardon Marbo 70.00
  2530 Napolitanke Bambi 230.00
 - Za kraj za kraj rada kase, pritisnite CTRL+D!
 - Za nov racun unesite kod artikla:
 1001
  Trazili ste: Keks Jaffa 120.00
 Unesite kod artikla [ili 0 za prekid]: 23
  Trazili ste: MedenoSrce Pionir 150.00
 Unesite kod artikla [ili 0 za prekid]: 0
  UKUPNO: 270.00 dinara
 - Za kraj za kraj rada kase, pritisnite CTRL+D!
 - Za nov racun unesite kod artikla:
  Greska: Ne postoji proizvod sa trazenim kodom!
 Unesite kod artikla [ili 0 za prekid]: 2530
  Trazili ste: Napolitanke Bambi 230.00
 Unesite kod artikla [ili 0 za prekid]: 0
  UKUPNO: 230.00 dinara.
 - Za kraj za kraj rada kase, pritisnite CTRL+D!
```

- Za nov racun unesite kod artikla:

Kraj rada kase!

Zadatak 3.24 Napisati program koji iz datoteke aktivnost.txt čita podatke o aktivnostima studenata na praktikumima i upisuje tri spiska redom u datoteke dat1.txt, dat2.txt i dat3.txt. Na prvom su studenti sortirani leksikografski po imenu rastuće. Na drugom su sortirani po ukupnom broju urađenih zadataka opadajuće, a ukoliko neki studenti imaju isti broj rešenih zadataka sortiraju se po dužini imena rastuće. Na trećem spisku kriterijum sortiranja je broj časova na kojima su bili, opadajuće. Ukoliko neki studenti imaju isti broj časova, sortirati ih opadajuće po broju urađenih zadataka, a ukoliko se i on poklapa sortirati po prezimenu opadajuće. U datoteci se nalazi ime, prezime studenta, broj

časova na kojima je prisustvovao, kao i ukupan broj urađenih zadataka. Pretpostaviti da studenata neće biti više od 500 i da je za ime studenta dovoljno 20, a za prezime 25 karaktera.

#### Test, 1

```
DAT2.TXT
AKTIVNOSTI.TXT
                                                    Studenti sortirani po broju zadataka
 Aleksandra Cvetic 4 6
 Bojan Golubovic 4 3
                                                    opadajuce, pa po duzini imena rastuce:
                                                    Aleksandra Cvetic 4 6
 Dragan Markovic 3 5
                                                    Uros Milic 2 5
 Ivana Stankovic 3 1
                                                    Dragan Markovic 3 5
 Marija Stankovic 1 3
 Ognjen Peric 1 2
                                                    Andrija Petrovic 2 5
 Uros Milic 2 5
                                                    Nenad Brankovic 2 4
                                                    Lazar Micic 1 3
 Andrija Petrovic 2 5
                                                    Bojan Golubovic 4 3
 Anja Ilic 3 1
 Lazar Micic 1 3
                                                    Marija Stankovic 1 3
                                                    Ognjen Peric 1 2
 Nenad Brankovic 2 4
                                                    Anja Ilic 3 1
                                                    Ivana Stankovic 3 1
DAT1.TXT
 Studenti sortirani po imenu
                                                  DAT3.TXT
 leksikografski rastuce:
                                                    Studenti sortirani po prisustvu
 Aleksandra Cvetic 4 6
                                                    opadajuce, pa po broju zadataka,
 Andrija Petrovic 2 5
 Anja Ilic 3 1
                                                    pa po prezimenima leksikografski
 Bojan Golubovic 4 3
                                                    opadajuce:
                                                    Aleksandra Cvetic 4 6
 Dragan Markovic 3 5
                                                    Bojan Golubovic 4 3
 Ivana Stankovic 3 1
 Lazar Micic 1 3
                                                    Dragan Markovic 3 5
                                                    Ivana Stankovic 3 1
 Marija Stankovic 1 3
                                                    Anja Ilic 3 1
 Nenad Brankovic 2 4
                                                    Andrija Petrovic 2 5
 Ognjen Peric 1 2
 Uros Milic 2 5
                                                    Uros Milic 2 5
                                                    Nenad Brankovic 2 4
                                                    Marija Stankovic 1 3
                                                    Lazar Micic 1 3
                                                    Ognjen Peric 1 2
```

Zadatak 3.25 U datoteci pesme.txt nalaze se informacije o gledanosti pesama na Youtube-u. Svaki red datoteke sadrži informacije o gledanosti pesama u formatu izvođač - naslov, broj gledanja. Napisati program koji učitava informacije o pesmama i vrši sortiranje pesama u zavisnosti od argumenata komandne linije na sledeći način:

- nema opcija, sortiranje se vrši po broju gledanja;
- prisutna je opcija -i, sortiranje se vrši po imenima izvođača;
- prisutna je opcija -n, sortiranje se vrši po naslovu pesama.

Na standardnom izlazu ispisati informacije o pesmama sortiranim na opisani način. Uraditi zadatak bez pravljenja pretpostavki o maksimalnoj dužini imena izvođača i naslova pesme.

```
Test 2
                                                              Test 3
Test 1
POKRETANJE: ./a.out
                               POKRETANJE: ./a.out -i
                                                              POKRETANJE: ./a.out -n
PESME.TXT
                               PESME.TXT
                                                              PESME.TXT
 Ana - Nebo, 2342
                                Ana - Nebo, 2342
                                                                Ana - Nebo, 2342
 Laza - Oblaci, 29
                                Laza - Oblaci, 29
                                                                Laza - Oblaci, 29
 Pera - Ptice, 327
                                Pera - Ptice, 327
                                                                Pera - Ptice, 327
 Jelena - Sunce, 92321
                                Jelena - Sunce, 92321
                                                                Jelena - Sunce, 92321
 Mika - Kisa, 5341
                                Mika - Kisa, 5341
                                                                Mika - Kisa, 5341
TZLAZ:
                               IZLAZ:
                                                              IZLAZ:
 Jelena - Sunce, 92321
                                Ana - Nebo, 2342
                                                                Mika - Kisa, 5341
 Mika - Kisa, 5341
                                Jelena - Sunce, 92321
                                                                Ana - Nebo, 2342
 Ana - Nebo, 2342
                                Laza - Oblaci, 29
                                                                Laza - Oblaci, 29
 Pera - Ptice, 327
                                Mika - Kisa, 5341
                                                                Pera - Ptice, 327
 Laza - Oblaci, 29
                                Pera - Ptice, 327
                                                                Jelena - Sunce, 92321
```

\* Zadatak 3.26 Razmatrajmo dve operacije: operacija U je unos novog broja x, a operacija N određivanje n-tog po veličini od unetih brojeva. Implementirati program koji izvršava ove operacije. Može postojati najviše 100000 operacija unosa, a uneti elementi se mogu ponavljati, pri čemu se i ponavljanja računaju prilikom brojanja. Optimizovati program, ukoliko se zna da neće biti više od 500 različitih unetih brojeva. NAPOMENA: Brojeve čuvati u sortiranom nizu i svaki naredni element umetati na svoje mesto.

```
Primer 1

INTERAKCIJA SA PROGRAMOM:
Unesite niz operacija: U 2 U 0 U 6 U 4 N 1 U 8 N 2 N 5 U 2 N 3 N 5
0 2 8 2 6
```

\* Zadatak 3.27 Šef u restoranu je neuredan i palačinke koje ispeče ne slaže redom po veličini. Konobar pre serviranja mora da sortira palačinke po veličini, a jedina operacija koju sme da izvodi je da obrne deo palačinki. Na primer, slika 3.1 po kolonama predstavlja naslagane palačinke posle svakog okretanja. Na početku, palačinka veličine 2 je na dnu, iznad nje se redom nalaze najmanja, najveća, itd... Na slici crtica predstavlja mesto iznad koga će konobar okrenuti palačinke. Prvi potez konobara je okretanje palačinki veličine 5, 4 i 3 (prva kolona), i tada će veličine palačinki odozdo nagore biti 2, 1, 3, 4, 5 (druga kolona). Posle još dva okretanja, palačinke će biti složene. Napisati program koji u najviše 2n-3 okretanja sortira učitani niz. UPUTSTVO: Imitirati selection sort i u svakom koraku dovesti jednu palačinku na svoje mesto korišćenjem najviše dva okretanja.

3	5	2	1
4 5_	4	1_	2
5_	3	1_ 3	3
1	1	4	4
2	2	5	5

Slika 3.1: Zadatak 3.27

```
Test 1

ULAZ:
23 64 123 76 22 7 34 123 54562 12 453 342 5342 42 542 1 3 432 1 32 43

IZLAZ:
1 1 3 7 12 22 23 32 34 42 43 64 76 123 123 342 432 453 542 5342 54562
```

**Zadatak 3.28** Za zadatu celobrojnu matricu dimenzije  $n \times m$  napisati funkciju koja vrši sortiranje vrsta matrice rastuće na osnovu sume elemenata u vrsti. Napisati potom program koji testira ovu funkciju. Sa standardnog ulaza se prvo unose dimenzije matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardnom izlazu. Napomena: Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

```
Test 1
                                                    Test 2
INTERAKCIJA SA PROGRAMOM:
                                                  INTERAKCIJA SA PROGRAMOM:
 Unesite dimenzije matrice: 3 2
                                                    Unesite dimenzije matrice: 4 4
 Unesite elemente matrice po vrstama:
                                                    Unesite elemente matrice po vrstama:
 6 -5
                                                    34 12 54 642
 -4 3
                                                    1234
 2 1
                                                    53 2 1 5
 Sortirana matrica je:
                                                    54 23 5 671
 -4 3
                                                    Sortirana matrica je:
 6 -5
                                                    1 2 3 4
 2 1
                                                    53 2 1 5
                                                    34 12 54 642
                                                    54 23 5 671
```

Zadatak 3.29 Za zadatu kvadratnu matricu dimenzije n napisati funkciju koja sortira kolone matrice opadajuće na osnovu vrednosti prvog elementa u koloni. Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unosi dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardnom izlazu. Napomena: Koristiti biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

# Primer 1 | Interakcija sa programom: | Unesite dimenziju matrice: 2 | Unesite elemente matrice po vrstama: 6 -5 | -4 3 | | Sortirana matrica je: -5 6 | | 3 -4

```
INTERAKCIJA SA PROGRAMOM:
Unesite dimenziju matrice: 4
Unesite elemente matrice po vrstama:
34 12 54 642
1 2 3 4
53 2 1 5
54 23 5 671
Sortirana matrica je:
12 34 54 642
2 1 3 4
2 53 1 5
```

Primer 2

23 54 5 671

# 3.3 Bibliotečke funkcije pretrage i sortiranja

Zadatak 3.30 Napisati program u kome se prvo inicijalizuje statički niz struktura osoba sa članovima ime i prezime, a zatim se učitava jedan karakter i pronalazi i štampa jedna struktura iz niza osoba čije prezime počinje tim karakterom. Ako takva osoba ne postoji, štampati -1 na standardnom izlazu za greške. Niz struktura ima manje od 100 elemenata i uređen je u rastućem leksikografskom poretku po prezimenima. Pretaživanje niza vršiti bibliotečkom funkcijom bsearch. Na primer, niz osoba može da bude inicijalizovan na sledeći način:



Zadatak 3.31 Napisati program koji ilustruje upotrebu bibliotečkih funkcija za pretraživanje i sortiranje nizova i mogućnost zadavanja različitih kriterijuma sortiranja. Sa standardnog ulaza se unosi dimenzija niza celih brojeva, ne veća od 100, a potom i sami elementi niza. Upotrebom funkcije qsort sortirati niz u rastućem poretku, sa standardnog ulaza učitati broj koji se traži u nizu, pa

zatim funkcijama bsearch i lfind utvrditi da li se zadati broj nalazi u nizu. Na standardnom izlazu ispisati odgovarajuću poruku.

```
Primer 1
                                                   Primer 2
INTERAKCIJA SA PROGRAMOM:
                                                  INTERAKCIJA SA PROGRAMOM:
 Uneti dimenziju niza: 11
                                                   Uneti dimenziju niza: 4
                                                   Uneti elemente niza:
 Uneti elemente niza:
 5 3 1 6 8 90 34 5 3 432 34
                                                   4257
 Sortirani niz u rastucem poretku:
                                                   Sortirani niz u rastucem poretku:
 1 3 3 5 5 6 8 34 34 90 432
                                                   2 4 5 7
 Uneti element koji se trazi u nizu: 34
                                                   Uneti element koji se trazi u nizu: 3
 Binarna pretraga:
                                                   Binarna pretraga:
 Element je nadjen na poziciji 8
                                                   Elementa nema u nizu!
 Linearna pretraga (lfind):
                                                   Linearna pretraga (lfind):
 Element je nadjen na poziciji 7
                                                   Elementa nema u nizu!
```

Zadatak 3.32 Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva (ne veću od 100), a potom i same elemente niza. Upotrebom funkcije qsort sortirati niz u rastućem poretku prema broju delilaca i tako dobijeni niz odštampati na standardnom izlazu.

```
Primer 1
                              Primer 2
                                                             Primer 3
INTERAKCIJA SA PROGRAMOM:
                             INTERAKCIJA SA PROGRAMOM:
                                                            INTERAKCIJA SA PROGRAMOM:
 Uneti dimenziju niza: 10
                                Uneti dimenziju niza: 1
                                                               Uneti dimenziju niza: 0
 Uneti elemente niza:
                                Uneti elemente niza:
                                                               Uneti elemente niza:
 1 2 3 4 5 6 7 8 9 10
                                234
                                                              Sortirani niz u rastucem
 Sortirani niz u rastucem
                                Sortirani niz u rastucem
                                                              poretku prema broju
                               poretku prema broju delilaca
 poretku prema broju delilaca
                                                               delilaca:
 1 2 3 5 7 4 9 6 8 10
                               234
```

Zadatak 3.33 Korišćenjem bibliotečke funkcije qsort napisati program koji sortira niz niski po sledećim kriterijumima:

- (a) leksikografski,
- (b) po dužini.

Niske se učitavaju iz datoteke niske.txt. Pretpostaviti da datoteka ne sadrži više od 1000 niski kao i da je svaka niska dužine najviše 30 karaktera. Program prvo leksikografski sortira niz, primenjuje binarnu pretragu (bsearch) zarad traženja niske unete sa standardnog ulaza, a potom traži istu nisku koristeći funkciju lfind u nizu koji je neposredno pre toga sortiran po dužini. Rezultate svih sortiranja i pretraga ispisati na standardnom izlazu.

#### Primer 1

```
NISKE.TXT
ana petar andjela milos nikola aleksandar ljubica matej milica

INTERAKCIJA SA PROGRAMOM:
Leksikografski sortirane niske:
aleksandar ana andjela ljubica matej milica milos nikola petar
Uneti trazenu nisku: matej
Niska "matej" je pronadjena u nizu na poziciji 4
Niske sortirane po duzini:
ana matej milos petar milica nikola andjela ljubica aleksandar
Niska "matej" je pronadjena u nizu na poziciji 1
```

Zadatak 3.34 Uraditi zadatak 3.33 sa dinamički alociranim niskama i sortiranjem niza pokazivača, umesto niza niski.

Zadatak 3.35 Napisati program koji korišćenjem bibliotečke funkcije qsort sortira studente prema broju poena osvojenih na kolokvijumu. Ukoliko više studenata ima isti broj bodova, sortirati ih po prezimenu leksikografski rastuće. Korisnik potom unosi broj bodova i prikazuje mu se jedan od studenata sa tim brojem bodova ili poruka ukoliko nema takvog. Potom, sa standardnog ulaza, unosi se prezime traženog studenta i prikazuje se osoba sa tim prezimenom ili poruka da se nijedan student tako ne preziva. Za pretraživanje koristiti odgovarajuće bibliotečke funkcije. Podaci o studentima čitaju se iz datoteke čije se ime zadaje preko argumenata komandne linije. Za svakog studenta u datoteci postoje ime, prezime i bodovi osvojeni na kolokvijumu. Pretpostaviti da neće biti više od 500 studenata i da je za ime i prezime svakog studenta dovoljno po 20 karaktera.

#### Primer 1

```
POKRETANJE: ./a.out kolokvijum.txt

ULAZNA DATOTEKA (KOLOKVIJUM.TXT):

Aleksandra Cvetic 15
Bojan Golubovic 30
Dragan Markovic 25
Filip Dukic 20
Ivana Stankovic 25
Marija Stankovic 15
Ognjen Peric 20
Uros Milic 10
Andrija Petrovic 0
Anja Ilic 5
Ivana Markovic 5
Lazar Micic 20
Nenad Brankovic 15
```

```
INTERAKCIJA SA PROGRAMOM:
 Studenti sortirani po broju poena
 opadajuce, pa po prezimenu rastuce:
 Bojan Golubovic 30
 Dragan Markovic 25
 Ivana Stankovic 25
 Filip Dukic 20
 Lazar Micic 20
 Ognjen Peric 20
 Nenad Brankovic 15
 Aleksandra Cvetic 15
 Marija Stankovic 15
 Uros Milic 10
 Anja Ilic 5
 Ivana Markovic 5
 Andrija Petrovic 0
 Unesite broj bodova: 20
 Pronadjen je student sa unetim
 brojem bodova: Filip Dukic 20
 Unesite prezime: Markovic
 Pronadjen je student sa unetim
 prezimenom: Dragan Markovic 25
```

Zadatak 3.36 Uraditi zadatak 3.13, ali korišćenjem bibliotečke qsort funkcije.

**Zadatak 3.37** Napisati program koji sa standardnog ulaza učitava prvo ceo broj  $n \ (n \le 10)$ , a zatim niz S od n niski. Maksimalna dužina svake niske je 31 karakter. Sortirati niz S bibliotečkom funkcijom **qsort** i proveriti da li u njemu ima identičnih niski.

```
Primer 2
                                                            Primer 3
Primer 1
                            INTERAKCIJA SA PROGRAMOM:
INTERAKCIJA SA PROGRAMOM:
                                                           INTERAKCIJA SA PROGRAMOM:
 Unesite broj niski: 4
                               Unesite broj niski: 3
                                                             Unesite broj niski: 5
 Unesite niske:
                               Unesite niske:
                                                             Unesite niske:
 prog search sort search
                               test kol ispit
                                                             a ab abc abcd abcde
 ima
                               nema
                                                             nema
```

Zadatak 3.38 Datoteka studenti.txt sadrži spisak studenata. Za svakog studenta poznat je nalog na Alas-u (oblika npr. mr15125, mm14001), ime, prezime i broj poena. Ni ime, ni prezime, neće biti duže od 20 karaktera. Napisati program koji korišćenjem funkcije qsort sortira studente po broju poena opadajuće, ukoliko je prisutna opcija -p, ili po nalogu, ukoliko je prisutna opcija -n. Studenti se po nalogu sortiraju tako što se sortiraju na osnovu godine, zatim na osnovu smera, i na kraju na osnovu broja indeksa. Sortirane studente upisati u datoteku izlaz.txt. Ukoliko je u komandnoj liniji uz opciju -n naveden i nalog nekog studenta, funkcijom bsearch potražiti i prijaviti broj poena studenta sa tim nalogom.

```
Test 2
 Test. 1
POKRETANJE: ./a.out -n mm13321
                                                  POKRETANJE: /a.out -p
STUDENTI.TXT
                                                  STUDENTI.TXT
 mr14123 Marko Antic 20
                                                   mr14123 Marko Antic 20
 mm13321 Marija Radic 12
                                                   mm13321 Marija Radic 12
 ml13011 Ivana Mitrovic 19
                                                   ml13011 Ivana Mitrovic 19
 ml13066 Pera Simic 15
                                                   ml13066 Pera Simic 15
 mv14003 Jovan Jovanovic 17
                                                   mv14003 Jovan Jovanovic 17
IZLAZ, TXT
                                                  IZLAZ, TXT
 ml13011 Ivana Mitrovic 19
                                                   mr14123 Marko Antic 20
 ml13066 Pera Simic 15
                                                   ml13011 Ivana Mitrovic 19
 mm13321 Marija Radic 12
                                                   mv14003 Jovan Jovanovic 17
 mr14123 Marko Antic 20
                                                    ml13066 Pera Simic 15
 mv14003 Jovan Jovanovic 17
                                                   mm13321 Marija Radic 12
 mm13321 Marija Radic 12
```

Zadatak 3.39 Definisati strukturu Datum. Napisati funkciju koja poredi dva

datuma hronološki. Potom, napisati i program koji učitava datume iz datoteke koja se zadaje kao prvi argument komandne linije (ne više od 128 datuma), sortira ih pozivajući funkciju qsort iz standardne biblioteke i pozivanjem funkcije bsearch iz standardne biblioteke proverava da li datumi učitani sa standardnog ulaza postoje među prethodno unetim datumima. Datumi se učitavaju sve do kraja ulaza.

#### Primer 1

# 3.4 Rešenja

```
#include <stdio.h>
  #include <stdlib.h>
  #include <time.h>
  #define MAX 1000000
  /* Pri prevodjenju program linkovati sa bibliotekom librt opcijom
     -lrt zbog funkcije clock_gettime() */
  /* Naredne tri funkcije koje vrse pretragu, ukoliko se trazeni
     element pronadje u nizu, vracaju indeks pozicije na kojoj je
     element pronadjen. Ovaj indeks je uvek nenegativan. Ako element
     nije pronadjen u nizu, funkcije vracaju negativnu vrednost -1,
     kao indikator neuspesne pretrage. */
  /* Linearna pretraga: Funkcija pretrazuje niz a[] celih brojeva
     duzine n, trazeci u njemu prvo pojavljivanje elementa x. Pretraga
    se vrsi prostom iteracijom kroz niz. */
  int linearna_pretraga(int a[], int n, int x)
19 {
    int i:
   for (i = 0; i < n; i++)
21
      if (a[i] == x)
        return i;
    return -1;
25 }
```

```
27 /* Binarna pretraga: Funkcija trazi u sortiranom nizu a[] duzine n
     broj x. Pretraga koristi osobinu sortiranosti niza i u svakoj
     iteraciji polovi interval pretrage. */
  int binarna_pretraga(int a[], int n, int x)
  {
31
    int levi = 0;
    int desni = n - 1;
33
    int srednji;
    /* Dokle god je indeks levi levo od indeksa desni */
    while (levi <= desni) {
      /* Srednji indeks je njihova aritmeticka sredina */
      srednji = (levi + desni) / 2;
      /* Ako je element sa sredisnjim indeksom veci od x, tada se x
39
         mora nalaziti u levom delu niza */
      if (x < a[srednji])</pre>
41
        desni = srednji - 1;
      /* Ako je element sa sredisnjim indeksom manji od x, tada se x
43
         mora nalaziti u desnom delu niza */
      else if (x > a[srednji])
45
        levi = srednji + 1;
      else
        /* Ako je element sa sredisnjim indeksom jednak x, tada je
           broj x pronadjen na poziciji srednji */
49
        return srednji;
    /* Ako element x nije pronadjen, vraca se -1 */
    return -1;
53
  /* Interpolaciona pretraga: Funkcija trazi u sortiranom nizu a[]
     duzine n broj x. Pretraga koristi osobinu sortiranosti niza i
     zasniva se na linearnoj interpolaciji vrednosti koja se trazi
     vrednostima na krajevima prostora pretrage. */
59
  int interpolaciona_pretraga(int a[], int n, int x)
61
    int levi = 0;
    int desni = n - 1;
63
    int srednji;
    /* Dokle god je indeks levi levo od indeksa desni... */
    while (levi <= desni) {
      /* Ako je trazeni element manji od pocetnog ili veci od
         poslednjeg elementa u delu niza a[levi],...,a[desni], tada on
         nije u tom delu niza. Ova provera je neophodna, da se ne bi
         dogodilo da se prilikom izracunavanja indeksa srednji izadje
         izvan opsega indeksa [levi,desni] */
      if (x < a[levi] || x > a[desni])
        return -1;
73
      /* U suprotnom, x je izmedju a[levi] i a[desni], pa ako su
         a[levi] i a[desni] jednaki, tada je jasno da je trazeni broj
         x jednak ovim vrednostima, pa se vraca indeks levi (ili
         indeks desni). Ova provera je neophodna, jer bi se u
         suprotnom prilikom izracunavanja indeksa srednji pojavilo
```

```
79
          deljenje nulom. */
       else if (a[levi] == a[desni])
         return levi;
81
       /* Racuna se sredisnji indeks */
       srednji =
83
           levi +
           (int) ((double) (x - a[levi]) / (a[desni] - a[levi]) *
85
                  (desni - levi)):
       /* Napomena: Indeks srednji je uvek izmedju levi i desni, ali ce
87
          verovatno biti blize trazenoj vrednosti nego da je prosto uvek
          uzimana aritmiticka sredina indeksa levi i desni. Ovo se moze
89
          porediti sa pretragom recnika: ako neko trazi rec na slovo
          'B', sigurno nece da otvori recnik na polovini, vec verovatno
          negde blize pocetku. */
       /* Ako je element sa indeksom srednji veci od trazenog, tada se
93
          trazeni element mora nalaziti u levoj polovini niza */
       if (x < a[srednji])</pre>
95
         desni = srednji - 1;
       /* Ako je element sa indeksom srednji manji od trazenog, tada se
          trazeni element mora nalaziti u desnoj polovini niza */
       else if (x > a[srednji])
99
         levi = srednji + 1;
       else
         /* Ako je element sa indeksom srednji jednak trazenom, onda se
            pretraga zavrsava na poziciji srednji */
         return srednji;
     /* U slucaju neuspesne pretrage vraca se -1 */
     return -1;
   int main(int argc, char **argv)
111 {
     int a[MAX];
113
     int n, i, x;
     struct timespec vreme1, vreme2, vreme3, vreme4, vreme5, vreme6;
     FILE *f;
     /* Provera argumenata komandne linije */
     if (argc != 3) {
117
       fprintf(stderr,
                "Greska: Program se poziva sa %s dim_niza broj\n",
119
               argv[0]);
       exit(EXIT_FAILURE);
121
     /* Dimenzija niza */
     n = atoi(argv[1]);
     if (n > MAX || n \le 0) {
       fprintf(stderr, "Greska: Dimenzija niza neodgovarajuca\n");
       exit(EXIT_FAILURE);
     }
129
```

```
/* Broj koji se trazi */
x = atoi(argv[2]);
/* Elementi niza se generisu slucajno, tako da je svaki sledeci
   veci od prethodnog. Funkcija srandom() inicijalizuje pocetnu
   vrednost sa kojom se krece u izracunavanje sekvence
   pseudo-slucajnih brojeva. Kako generisani niz ne bi uvek bio
   isti, ova vrednost se postavlja na tekuce vreme u sekundama od
   Nove godine 1970, tako da je za svako sledece pokretanje
   programa (u vremenskim intervalima vecim od jedne sekunde) ove
   vrednost drugacija. random()%100 vraca brojeve izmedju 0 i 99 */
srandom(time(NULL));
for (i = 0; i < n; i++)
 a[i] = i == 0 ? random() % 100 : a[i - 1] + random() % 100;
/* Lineara pretraga */
printf("Linearna pretraga:\n");
/* Vreme proteklo od Nove godine 1970 */
clock_gettime(CLOCK_REALTIME, &vreme1);
i = linearna_pretraga(a, n, x);
/* Novo vreme i razlika sa prvim predstavlja vreme utroseno za
  linearnu pretragu */
clock_gettime(CLOCK_REALTIME, &vreme2);
if (i == -1)
 printf("Element nije u nizu\n");
else
 printf("Element je u nizu na poziciji %d\n", i);
/* Binarna pretraga */
printf("Binarna pretraga:\n");
clock_gettime(CLOCK_REALTIME, &vreme3);
i = binarna_pretraga(a, n, x);
clock_gettime(CLOCK_REALTIME, &vreme4);
if (i == -1)
 printf("Element nije u nizu\n");
else
 printf("Element je u nizu na poziciji %d\n", i);
/* Interpolaciona pretraga */
printf("Interpolaciona pretraga:\n");
clock_gettime(CLOCK_REALTIME, &vreme5);
i = interpolaciona_pretraga(a, n, x);
clock_gettime(CLOCK_REALTIME, &vreme6);
if (i == -1)
 printf("Element nije u nizu\n");
else
 printf("Element je u nizu na poziciji %d\n", i);
/* Podaci o izvrsavanju programa bivaju upisani u log */
if ((f = fopen("vremena.txt", "a")) == NULL) {
 fprintf(stderr, "Greska: Neuspesno otvaranje log datoteke.\n");
  exit(EXIT_FAILURE);
fprintf(f, "Dimenzija niza: %d\n", n);
fprintf(f, "\tLinearna:%10ld ns\n",
        (vreme2.tv_sec - vreme1.tv_sec) * 1000000000 +
```

```
#include <stdio.h>
2 #include <stdlib.h>
  #define MAX 1024
6 /* Rekurzivna linearna pretraga od pocetka niza */
  int linearna_pretraga_r1(int a[], int n, int x)
    int tmp;
    /* Izlaz iz rekurzije */
    if (n \le 0)
     return -1;
12
    /* Ako je prvi element trazeni */
    if (a[0] == x)
     return 0;
16
    /* Pretraga ostatka niza */
    tmp = linearna_pretraga_r1(a + 1, n - 1, x);
    return tmp < 0 ? tmp : tmp + 1;
18
  }
20
  /* Rekurzivna linearna pretraga od kraja niza */
22 int linearna_pretraga_r2(int a[], int n, int x)
    /* Izlaz iz rekurzije */
24
    if (n \le 0)
26
     return -1;
    /* Ako je poslednji element trazeni */
    if (a[n - 1] == x)
28
      return n - 1;
    /* Pretraga ostatka niza */
30
    return linearna_pretraga_r2(a, n - 1, x);
32 }
34 /* Rekurzivna binarna pretraga */
  int binarna_pretraga_r(int a[], int 1, int d, int x)
36 {
```

```
int srednji;
    if (1 > d)
38
      return -1;
    /* Sredisnja pozicija na kojoj se trazi vrednost x */
40
    srednji = (1 + d) / 2;
    /* Ako je element na sredisnjoj poziciji trazeni */
42
    if (a[srednji] == x)
      return srednji;
44
    /* Ako je trazeni broj veci od broja na sredisnjoj poziciji,
       pretrazuje se desna polovina niza */
46
    if (a[srednji] < x)</pre>
48
      return binarna_pretraga_r(a, srednji + 1, d, x);
    /* Ako je trazeni broj manji od broja na sredisnjoj poziciji,
       pretrazuje se leva polovina niza */
      return binarna_pretraga_r(a, 1, srednji - 1, x);
54
  /* Rekurzivna interpolaciona pretaga */
int interpolaciona_pretraga_r(int a[], int 1, int d, int x)
    int p;
58
    /* Ako je trazeni element manji od prvog ili veci od poslednjeg */
    if (x < a[1] || x > a[d])
60
      return -1;
    /* Ako je ostao jedan element u delu niza koji se pretrazuje */
    if (a[d] == a[1])
      return 1;
64
    /* Pozicija na kojoj se trazi vrednost x */
    p = 1 + (d - 1) * (x - a[1]) / (a[d] - a[1]);
    if (a[p] == x)
      return p;
68
    /* Pretraga sufiksa niza */
    if (a[p] < x)
      return interpolaciona_pretraga_r(a, p + 1, d, x);
    /* Pretraga prefiksa niza */
    else
      return interpolaciona_pretraga_r(a, l, p - 1, x);
74
76
  int main()
78
    int a[MAX];
80
    int x;
    int i, indeks;
82
    /* Ucitava se trazeni broj */
    printf("Unesite trazeni broj: ");
    scanf("%d", &x);
86
    /* Ucitavaju se elemenati niza sve do kraja ulaza - ocekuje se da
       korisnik pritisne CTRL+D za naznaku kraja */
```

```
i = 0;
     printf("Unesite sortiran niz elemenata: ");
90
     while (i < MAX && scanf("%d", &a[i]) == 1) {
       if (i > 0 \&\& a[i] < a[i - 1]) {
         fprintf(stderr, "Greska: Elementi moraju biti uneseni ");
         fprintf(stderr, "u neopadajucem poretku\n");
         exit(EXIT_FAILURE);
96
       i++;
     }
98
     /* Rezultati linearnih pretraga */
100
     printf("Linearna pretraga\n");
     indeks = linearna_pretraga_r1(a, i, x);
     if (indeks == -1)
       printf("Element se ne nalazi u nizu.\n");
104
     else
       printf("Pozicija elementa je %d.\n", indeks);
106
     indeks = linearna_pretraga_r2(a, i, x);
     if (indeks == -1)
108
       printf("Element se ne nalazi u nizu.\n");
     else
       printf("Pozicija elementa je %d.\n", indeks);
112
     /* Rezultati binarna pretrage */
     printf("Binarna pretraga\n");
114
     indeks = binarna_pretraga_r(a, 0, i - 1, x);
     if (indeks == -1)
       printf("Element se ne nalazi u nizu.\n");
     else
118
       printf("Pozicija elementa je %d.\n", indeks);
120
     /* Rezultati interpolacione pretrage */
     printf("Interpolaciona pretraga\n");
     indeks = interpolaciona_pretraga_r(a, 0, i - 1, x);
     if (indeks == -1)
124
       printf("Element se ne nalazi u nizu.\n");
     else
       printf("Pozicija elementa je %d.\n", indeks);
128
     exit(EXIT_SUCCESS);
130 }
```

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

#define MAX_STUDENATA 128
6 #define MAX_DUZINA 16
```

```
/* O svakom studentu postoje 3 informacije i one su objedinjene u
     strukturi kojom se predstavlja svaki student. */
10 typedef struct {
    /* Indeks mora biti tipa long jer su podaci u datoteci preveliki
       za int, npr. 20140123 */
    long indeks;
    char ime[MAX_DUZINA];
14
    char prezime[MAX_DUZINA];
16 } Student;
  /* Ucitan niz studenata ce biti sortiran rastuce prema indeksu, jer
     su studenti u datoteci vec sortirani. Iz tog razloga pretraga po
     indeksu se vrsi binarno, dok pretraga po prezimenu mora linearno,
     jer nema garancije da postoji uredjenje po prezimenu. */
  /* Funkcija trazi u sortiranom nizu studenata a[] duzine n studenta
     sa indeksom x i vraca indeks pozicije nadjenog clana niza ili -1,
24
     ako element nije pronadjen. */
  int binarna_pretraga(Student a[], int n, long x)
26
    int levi = 0;
28
    int desni = n - 1;
    int srednji;
30
    /* Dokle god je indeks levi levo od indeksa desni */
    while (levi <= desni) {
      /* Racuna se srednja pozicija */
      srednji = (levi + desni) / 2;
34
      /* Ako je indeks stutenta na toj poziciji veci od trazenog, tada
         se trazeni indeks mora nalaziti u levoj polovini niza */
36
      if (x < a[srednji].indeks)</pre>
        desni = srednji - 1;
38
      /* Ako je pak manji od trazenog, tada se on mora nalaziti u
         desnoj polovini niza */
40
      else if (x > a[srednji].indeks)
        levi = srednji + 1;
42
      else
        /* Ako je jednak trazenom indeksu x, tada je pronadjen student
44
            sa trazenom indeksom na poziciji srednji */
        return srednji;
46
    /* Ako nije pronadjen, vraca se -1 */
48
    return -1;
50
  /* Linearnom pretragom niza studenata trazi se prezime x */
  int linearna_pretraga(Student a[], int n, char x[])
    int i;
    for (i = 0; i < n; i++)
56
      /* Poredi se prezimene i-tog studenta i poslatog x */
      if (strcmp(a[i].prezime, x) == 0)
```

```
return i;
    return -1;
62
   /* Main funkcija mora imati argumente jer se ime datoteke i opcija
     prosledjuju kao argumenti komandne linije */
   int main(int argc, char *argv[])
66 {
    Student dosije[MAX_STUDENATA];
   FILE *fin = NULL;
68
    int i;
    int br_studenata = 0;
    long trazen_indeks = 0;
    char trazeno_prezime[MAX_DUZINA];
72
    int bin_pretraga;
74
     /* Provera da li je korisnik prilikom poziva programa prosledio
       ime datoteke sa informacijama o studentima i opciju pretrage */
     if (argc != 3) {
      fprintf(stderr,
               "Greska: Program se poziva sa %s ime_datoteke opcija\n",
               argv[0]);
80
       exit(EXIT_FAILURE);
82
     /* Provera prosledjene opcije */
84
     if (strcmp(argv[2], "-indeks") == 0)
      bin_pretraga = 1;
86
     else if (strcmp(argv[2], "-prezime") == 0)
      bin_pretraga = 0;
88
     else {
90
       fprintf(stderr,
               "Greska: Opcija mora biti -indeks ili -prezime\n");
       exit(EXIT_FAILURE);
     }
94
     /* Otvara se datoteka */
     fin = fopen(argv[1], "r");
96
     if (fin == NULL) {
       fprintf(stderr,
98
               "Greska: Neuspesno otvaranje datoteke %s za citanje\n",
100
               argv[1]);
       exit(EXIT_FAILURE);
     }
     /* Cita se sve dok postoji red sa informacijama o studentu */
104
     i = 0;
     while (1) {
106
       if (i == MAX_STUDENATA)
        break;
108
       if (fscanf
           (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
110
```

```
dosije[i].prezime) != 3)
         break;
       i++:
114
     br_studenata = i;
     /* Nakon citanja, datoteka vise nije neophodna i zatvara se. */
     fclose(fin);
118
     /* Pretraga po indeksu */
120
     if (bin_pretraga) {
       /* Unos indeksa koji se binarno trazi u nizu */
       printf("Unesite indeks studenta cije informacije zelite: ");
       scanf("%ld", &trazen_indeks);
124
       i = binarna_pretraga(dosije, br_studenata, trazen_indeks);
       /* Rezultat binarne pretrage */
126
       if (i == -1)
         printf("Ne postoji student sa indeksom %ld\n", trazen_indeks);
128
       else
         printf("Indeks: %ld, Ime i prezime: %s %s\n",
130
                dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
     /* Pretraga po prezimenu */
     else {
134
       /* Unos prezimena koje se linearno trazi u nizu */
       printf("Unesite prezime studenta cije informacije zelite: ");
136
       scanf("%s", trazeno_prezime);
       i = linearna_pretraga(dosije, br_studenata, trazeno_prezime);
138
       /* Rezultat linearne pretrage */
       if (i == -1)
140
         printf("Ne postoji student sa prezimenom %s\n",
                trazeno_prezime);
142
         printf("Indeks: %ld, Ime i prezime: %s %s\n",
144
                 dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
146
     exit(EXIT_SUCCESS);
148
```

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

#define MAX_STUDENATA 128
#define MAX_DUZINA 16

typedef struct {
long indeks;
```

```
char ime[MAX_DUZINA];
    char prezime[MAX_DUZINA];
  } Student;
13
  int binarna_pretraga_rekurzivna(Student a[], int levi, int desni,
                                   long x)
    /* Ako je pozicija elementa na levom kraju veca od pozicije
       elementa na desnom kraju dela niza koji se pretrazuje, onda se
       zapravo pretrazuje prazan deo niza. U praznom delu niza nema
19
       trazenog elementa pa se vraca -1 */
    if (levi > desni)
     return -1;
    /* Racuna se pozicija srednjeg elementa */
    int srednji = (levi + desni) / 2;
    /* Da li je srednji bas onaj trazeni */
    if (a[srednji].indeks == x) {
     return srednji;
    /* Ako je trazeni indeks manji od indeksa studenta na srednjoj
29
       poziciji, onda se pretraga nastavlja u levoj polovini niza, jer
       je poznato da je niz sortiran po indeksu u rastucem poretku. */
31
    if (x < a[srednji].indeks)</pre>
      return binarna_pretraga_rekurzivna(a, levi, srednji - 1, x);
    /* Inace ga treba traziti u desnoj polovini */
      return binarna_pretraga_rekurzivna(a, srednji + 1, desni, x);
  }
37
39 int linearna_pretraga_rekurzivna_v2(Student a[], int n, char x[])
    /* Ako je niz prazan, vraca se -1 */
41
    if (n == 0)
     return -1;
43
    /* Kako se trazi prvi student sa trazenim prezimenom, pocinje se
       sa prvim studentom u nizu. */
45
    if (strcmp(a[0].prezime, x) == 0)
     return 0;
    int i = linearna_pretraga_rekurzivna_v2(a + 1, n - 1, x);
    return i >= 0 ? 1 + i : -1;
49
  int linearna_pretraga_rekurzivna(Student a[], int n, char x[])
53 \
    /* Ako je niz prazan, vraca se -1 */
    if (n == 0)
     return -1;
    /* Ako se trazi poslednji student sa trazenim prezimenom, pocinje
       se sa poslednjim studentom u nizu. */
59
    if (strcmp(a[n - 1].prezime, x) == 0)
     return n - 1;
    return linearna_pretraga_rekurzivna(a, n - 1, x);
```

```
/* Main funkcija mora imati argumente jer se ime datoteke i opcija
     prosledjuju kao argumenti komandne linije */
  int main(int argc, char *argv[])
  {
67
    Student dosije[MAX_STUDENATA];
    FILE *fin = NULL;
69
    int i;
    int br_studenata = 0;
    long trazen_indeks = 0;
    char trazeno_prezime[MAX_DUZINA];
73
    int bin_pretraga;
    /* Provera da li je korisnik prilikom poziva programa prosledio
       ime datoteke sa informacijama o studentima i opciju pretrage */
    if (argc != 3) {
      fprintf(stderr,
               "Greska: Program se poziva sa %s ime_datoteke opcija\n",
               argv[0]);
81
      exit(EXIT_FAILURE);
    }
83
    /* Provera prosledjene opcije */
85
    if (strcmp(argv[2], "-indeks") == 0)
      bin_pretraga = 1;
87
    else if (strcmp(argv[2], "-prezime") == 0)
      bin_pretraga = 0;
89
    else {
      fprintf(stderr,
91
               "Greska: Opcija mora biti -indeks ili -prezime\n");
      exit(EXIT_FAILURE);
93
95
    /* Otvara se datoteka */
    fin = fopen(argv[1], "r");
97
    if (fin == NULL) {
      fprintf(stderr,
99
               "Greska: Neuspesno otvaranje datoteke %s za citanje\n",
               argv[1]);
      exit(EXIT_FAILURE);
    /* Cita se sve dok postoji red sa informacijama o studentu */
    i = 0;
    while (1) {
      if (i == MAX_STUDENATA)
        break;
      if (fscanf
           (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
            dosije[i].prezime) != 3)
        break:
```

```
i++;
     }
     br_studenata = i;
117
     /* Nakon citanja, datoteka vise nije neophodna i zatvara se. */
     fclose(fin);
     /* Pretraga po indeksu */
     if (bin_pretraga) {
       /* Unos indeksa koji se binarno trazi u nizu */
123
       printf("Unesite indeks studenta cije informacije zelite: ");
       scanf("%ld", &trazen_indeks);
       i = binarna_pretraga_rekurzivna(dosije, 0, br_studenata,
                                        trazen_indeks);
       /* Rezultat binarne pretrage */
       if (i == -1)
         printf("Ne postoji student sa indeksom %ld\n", trazen_indeks);
       else
         printf("Indeks: %ld, Ime i prezime: %s %s\n",
                dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
133
     /* Pretraga po prezimenu */
135
     else {
       /* Unos prezimena koje se linearno trazi u nizu */
       printf("Unesite prezime studenta cije informacije zelite: ");
       scanf("%s", trazeno_prezime);
       i = linearna_pretraga_rekurzivna_v2(dosije, br_studenata,
                                            trazeno_prezime);
       /* Rezultat linearne pretrage */
       if (i == -1)
143
         printf("Ne postoji student sa prezimenom %s\n",
                trazeno_prezime);
145
         printf("Indeks: %ld, Ime i prezime: %s %s\n",
147
                dosije[i].indeks, dosije[i].ime, dosije[i].prezime);
     }
149
     exit(EXIT_SUCCESS);
```

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

#define MAX 1024

/* Struktura koja opisuje tacku u ravni */
typedef struct Tacka {
```

```
float x;
    float y;
  } Tacka;
13
  /* Funkcija koja racuna rastojanje zadate tacke od koordinatnog
    pocetka (0,0) */
  float rastojanje(Tacka A)
17 | {
    return sqrt(A.x * A.x + A.y * A.y);
19 }
  /* Funkcija koja pronalazi tacku najblizu koordinatnom pocetku u
     nizu zadatih tacaka t dimenzije n */
23 Tacka najbliza_koordinatnom(Tacka t[], int n)
    Tacka najbliza;
    int i;
    najbliza = t[0];
    for (i = 1; i < n; i++) {
      if (rastojanje(t[i]) < rastojanje(najbliza)) {</pre>
29
        najbliza = t[i];
31
    return najbliza;
33
35
  /* Funkcija koja pronalazi tacku najblizu x osi u nizu zadatih
     tacaka t dimenzije n */
  Tacka najbliza_x_osi(Tacka t[], int n)
39
    Tacka najbliza;
    int i;
41
    najbliza = t[0];
    for (i = 1; i < n; i++) {
43
      if (fabs(t[i].x) < fabs(najbliza.x)) {</pre>
        najbliza = t[i];
45
47
    return najbliza;
49
  /* Funkcija koja pronalazi tacku najblizu y osi u nizu zadatih
     tacaka t dimenzije n */
53 Tacka najbliza_y_osi(Tacka t[], int n)
    Tacka najbliza;
    int i;
    najbliza = t[0];
57
    for (i = 1; i < n; i++) {
      if (fabs(t[i].y) < fabs(najbliza.y)) {</pre>
59
        najbliza = t[i];
61
```

```
return najbliza;
   int main(int argc, char *argv[])
67 {
    FILE *ulaz;
    Tacka tacke[MAX];
    Tacka najbliza;
    int i, n;
71
    /* Ocekuje se da korisnik prosledi barem ime izvrsnog programa i
73
       ime datoteke sa tackama */
    if (argc < 2) {
      fprintf(stderr,
               "Greska: Programa se poziva sa %s ime_datoteke\n",
               argv[0]);
       exit(EXIT_FAILURE);
81
     /* Otvara se datoteka za citanje */
    ulaz = fopen(argv[1], "r");
83
     if (ulaz == NULL) {
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
85
               argv[1]);
       exit(EXIT_FAILURE);
87
    }
89
     /* Sve dok ima tacaka u datoteci, one bivaju smestane u niz sa
       tackama; i predstavlja indeks tekuce tacke */
91
    while (fscanf(ulaz, "%f %f", &tacke[i].x, &tacke[i].y) == 2) {
      i++;
    }
95
    n = i;
97
    /* Proverava se koji su dodatni argumenti komandne linije. Ako
       nema dodatnih argumenata */
99
     if (argc == 2)
      /* Trazi se najbliza tacka u odnosu na koordinatni pocetak */
      najbliza = najbliza_koordinatnom(tacke, n);
    /* Inace proverava se koji je dodatni argument prosledjen. Ako je
       u pitanju opcija -x */
     else if (strcmp(argv[2], "-x") == 0)
       /* Racuna se rastojanje u odnosu na x osu */
      najbliza = najbliza_x_osi(tacke, n);
     /* Ako je u pitanju opcija -y */
     else if (strcmp(argv[2], "-y") == 0)
       /* Racuna se rastojanje u odnosu na y osu */
      najbliza = najbliza_y_osi(tacke, n);
     else {
113
      /* Ako nije zadata opcija -x ili -y, ispisuje se obavestenje za
```

```
korisnika i prekida se izvrsavanje programa */
fprintf(stderr, "Greska: Pogresna opcija\n");
exit(EXIT_FAILURE);

}

/* Stampaju se koordinate trazene tacke */
printf("%g %g\n", najbliza.x, najbliza.y);

/* Zatvara se datoteka */
fclose(ulaz);

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
  #include <math.h>
  /* Tacnost */
  #define EPS 0.001
  int main()
    double 1, d, s;
10
    /* Kod intervala [0, 2] leva granica je 0, a desna 2 */
    1 = 0;
    d = 2;
    /* Sve dok se ne pronadje trazena vrednost argumenta */
    while (1) {
      /* Polovi se interval */
      s = (1 + d) / 2;
18
      /* Ako je apsolutna vrednost kosinusa u ovoj tacki manja od
         zadate tacnosti, prekida se pretraga */
      if (fabs(cos(s)) < EPS) {
        break;
      /* Ako je nula u levom delu intervala, nastavlja se pretraga na
         [1, s] */
      if (\cos(1) * \cos(s) < 0)
26
        d = s;
28
        /* Inace, na intervalu [s, d] */
        1 = s;
30
32
    /* Stampa se vrednost trazene tacke */
    printf("%g\n", s);
34
```

```
36 return 0; }
```

```
#include <stdio.h>
2 #include <stdlib.h>
  #include <string.h>
 #include <math.h>
6 int main(int argc, char **argv)
    double 1, d, s, epsilon;
    char ime_funkcije[6];
12
    /* Pokazivac na funkciju koja ima jedan argument tipa double i
       povratnu vrednost istog tipa */
    double (*fp) (double);
16
    /* Ako korisnik nije uneo argument, prijavljuje se greska */
    if (argc != 2) {
      fprintf(stderr, "Greska: Program se poziva sa %s ime_funkcije\n",
18
              argv[0]);
      exit(EXIT_FAILURE);
20
    }
    /* Niska ime_funkcije sadrzi ime trazene funkcije koja je navedena
      u komandnoj liniji */
    strcpy(ime_funkcije, argv[1]);
26
    /* Inicijalizuje se pokazivac na funkciju koja se tabelira */
    if (strcmp(ime_funkcije, "sin") == 0)
28
      fp = &sin;
    else if (strcmp(ime_funkcije, "cos") == 0)
30
      fp = &cos;
    else if (strcmp(ime_funkcije, "tan") == 0)
      fp = &tan;
    else if (strcmp(ime_funkcije, "atan") == 0)
34
      fp = &atan;
36
    else if (strcmp(ime_funkcije, "asin") == 0)
      fp = &asin;
    else if (strcmp(ime_funkcije, "log") == 0)
38
      fp = &log;
    else if (strcmp(ime_funkcije, "log10") == 0)
40
      fp = &log10;
    else {
42
      fprintf(stderr,
              "Greska: Program ne podrzava trazenu funkciju!\n");
44
      exit(EXIT_SUCCESS);
46
```

```
printf("Unesite krajeve intervala: ");
     scanf("%lf %lf", &1, &d);
50
     if ((*fp) (1) * (*fp) (d) >= 0) {
       fprintf(stderr, "Greska: %s na intervalu ", ime_funkcije);
fprintf(stderr, "[%g, %g] ne zadovoljava uslove\n", 1, d);
       exit(EXIT_FAILURE);
54
56
     printf("Unesite preciznost: ");
     scanf("%lf", &epsilon);
58
     /* Sve dok se ne pronadje trazena vrednost argumenta */
60
     while (1) {
       /* Polovi se interval */
62
       s = (1 + d) / 2;
       /* Ako je apsolutna vrednost trazene funkcije u ovoj tacki manja
64
          od zadate tacnosti, prekida se pretraga */
       if (fabs((*fp) (s)) < epsilon) {
66
         break;
       }
68
       /* Ako je nula u levom delu intervala, nastavlja se pretraga na
          [1, s] */
       if ((*fp) (1) * (*fp) (s) < 0)
         d = s;
       else
         /* Inace, na intervalu [s, d] */
         1 = s;
     }
     /* Stampa se vrednost trazene tacke */
    printf("%g\n", s);
80
     exit(EXIT_SUCCESS);
  }
82
```

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

#define MAX 256

int prvi_veci_od_nule(int niz[], int n)
{
    /* Granice pretrage */
    int l = 0, d = n - 1;
    int s;
    /* Sve dok je leva manja od desne granice */
    while (1 <= d) {</pre>
```

```
/* Racuna se sredisnja pozicija */
      s = (1 + d) / 2;
14
      /* Ako je broj na toj poziciji veci od nule, a eventualni njegov
         prethodnik manji ili jednak nuli, pretraga je zavrsena */
      if (niz[s] > 0 \&\& ((s > 0 \&\& niz[s - 1] <= 0) || s == 0))
        return s;
18
      /* U slucaju broja manjeg ili jednakog nuli, pretrazuje se desna
         polovina niza */
20
      if (niz[s] <= 0)
        1 = s + 1;
      /* A inace, leva polovina */
      else
24
        d = s - 1;
26
    return -1;
28
30 int main()
    int niz[MAX];
    int n = 0;
34
    /* Unos niza */
    while (scanf("%d", &niz[n]) == 1)
36
     n++;
38
    /* Stampa se rezultat */
    printf("%d\n", prvi_veci_od_nule(niz, n));
40
    return 0;
42
```

```
#include <stdio.h>
  #include <stdlib.h>
  #define MAX 256
5
  int prvi_manji_od_nule(int niz[], int n)
    /* Granice pretrage */
    int 1 = 0, d = n - 1;
    /* Sve dok je leva manja od desne granice */
    while (1 \le d) {
      /* Racuna se sredisnja pozicija */
13
      s = (1 + d) / 2;
      /* Ako je broj na toj poziciji manji od 0, a eventualni njegov
         prethodnik veci ili jednak 0, pretraga se zavrsava */
17
      if (niz[s] < 0 && ((s > 0 && niz[s - 1] >= 0) || s == 0))
```

```
return s;
       /* Ako je broj veci ili jednak 0, pretrazuje se desna polovina
19
          niza */
       if (niz[s] >= 0)
        1 = s + 1:
       /* A inace leva */
23
      else
         d = s - 1;
27
    return -1;
29
  int main()
  {
31
     int niz[MAX];
    int n = 0;
33
     /* Unos niza */
35
    while (scanf("%d", &niz[n]) == 1)
      n++;
37
    /* Stampa se rezultat */
39
    printf("%d\n", prvi_manji_od_nule(niz, n));
41
    return 0;
43 }
```

```
#include <stdio.h>
  #include <stdlib.h>
  unsigned int logaritam_a(unsigned int x)
5
    /* Izlaz iz rekurzije */
   if (x == 1)
      return 0;
    /* Rekurzivni korak */
    return 1 + logaritam_a(x >> 1);
11 }
unsigned int logaritam_b(unsigned int x)
    /* Binarnom pretragom se trazi jedinica u binarnom zapisu broja x
       najvece vaznosti, tj. najlevlja. Pretragu se vrsi od pozicije 0
17
       do 31 */
    int d = 0, l = sizeof(unsigned int) * 8 - 1;
    /* Sve dok je desna granica pretrage desnije od leve */
    while (d <= 1) {
21
      /* Racuna se sredisnja pozicija */
```

```
23
      s = (1 + d) / 2;
      /* Proverava se da li je na toj poziciji trazena jedinica */
      if ((1 << s) <= x && (1 << (s + 1)) > x)
        return s;
      /* Pretraga desne polovine binarnog zapisa */
      if ((1 << s) > x)
        1 = s - 1;
29
      /* Pretraga leve polovine binarnog zapisa */
      else
        d = s + 1;
    return s;
  }
35
37 int main()
    unsigned int x;
39
    /* Unos podatka */
41
    scanf("%u", &x);
43
    /* Provera da li je uneti broj pozitivan */
    if (x == 0) {
45
      fprintf(stderr, "Greska: Logaritam od nule nije definisan\n");
      exit(EXIT_FAILURE);
47
49
    /* Ispis povratnih vrednosti funkcija */
    printf("%u %u\n", logaritam_a(x), logaritam_b(x));
    exit(EXIT_SUCCESS);
```

#### sort.h

```
#ifndef _SORT_H_
#define _SORT_H_ 1

/* Selection sort: Funkcija sortira niz celih brojeva metodom
    sortiranja izborom. Ideja algoritma je sledeca: U svakoj
    iteraciji pronalazi se najmanji element i premesta se na pocetak
    niza. Dakle, u prvoj iteraciji, pronalazi se najmanji element, i
    dovodi na nulto mesto u nizu. U i-toj iteraciji najmanjih i-1
    elemenata su vec na svojim pozicijama, pa se od elemenata sa
    indeksima od i do n-1 trazi najmanji, koji se dovodi na i-tu
    poziciju. */
void selection_sort(int a[], int n);
```

```
14 /* Insertion sort: Funkcija sortira niz celih brojeva metodom
     sortiranja umetanjem. Ideja algoritma je sledeca: neka je na
     pocetku i-te iteracije niz prvih i elemenata
     (a[0],a[1],\ldots,a[i-1]) sortirano. U i-toj iteraciji treba element
     a[i] umetnuti na pravu poziciju medju prvih i elemenata tako da se
18
     dobije niz duzine i+1 koji je sortiran. Ovo se radi tako sto se
     i-ti element najpre uporedi sa njegovim prvim levim susedom
20
     (a[i-1]). Ako je a[i] vece, tada je on vec na pravom mestu, i niz
     a[0],a[1],...,a[i] je sortiran, pa se moze preci na sledecu
     iteraciju. Ako je a[i-1] vece, tada se zamenjuju a[i] i a[i-1], a
     zatim se proverava da li je potrebno dalje potiskivanje elementa u
24
     levo, poredeci ga sa njegovim novim levim susedom. Ovim uzastopnim
     premestanjem se a[i] umece na pravo mesto u nizu. */
26
  void insertion_sort(int a[], int n);
28
  /* Bubble sort: Funkcija sortira niz celih brojeva metodom mehurica.
     Ideja algoritma je sledeca: prolazi se kroz niz redom poredeci
30
     susedne elemente, i pri tom ih zamenjujuci ako su u pogresnom
     poretku. Ovim se najveci element poput mehurica istiskuje na
      "povrsinu", tj. na krajnju desnu poziciju. Nakon toga je potrebno
     ovaj postupak ponoviti nad nizom a[0],\ldots,a[n-2], tj. nad prvih
34
     n-1 elemenata niza bez poslednjeg koji je postavljen na pravu
     poziciju. Nakon toga se isti postupak ponavlja nad sve kracim i
36
     kracim prefiksima niza, cime se jedan po jedan istiskuju
     elemenenti na svoje prave pozicije. */
  void bubble_sort(int a[], int n);
40
  /* Selsort: Ovaj algoritam je jednostavno prosirenje sortiranja
     umetanjem koje dopusta direktnu razmenu udaljenih elemenata.
42
     Prosirenje se sastoji u tome da se kroz algoritam umetanja
     prolazi vise puta; u prvom prolazu, umesto koraka 1 uzima se neki
44
     korak h koji je manji od n (sto omogucuje razmenu udaljenih
     elemenata) i tako se dobija h-sortiran niz, tj. niz u kome su
46
     elementi na rastojanju h sortirani, mada susedni elementi to ne
     moraju biti. U drugom prolazu kroz isti algoritam sprovodi se
     isti postupak ali za manji korak h. Sa prolazima se nastavlja sve
50
     do koraka h = 1, u kome se dobija potpuno sortirani niz. Izbor
     pocetne vrednosti za h, i nacina njegovog smanjivanja menja u
     nekim slucajevima brzinu algoritma, ali bilo koja vrednost ce
     rezultovati ispravnim sortiranjem, pod uslovom da je u poslednjoj
     iteraciji h imalo vrednost 1. */
  void shell_sort(int a[], int n);
  /* Merge sort: Funkcija sortira niz celih brojeva a[] ucesljavanjem.
     Sortiranje se vrsi od elementa na poziciji 1 do onog na poziciji
58
     d. Na pocetku, da bi niz bio kompletno sortiran, 1 mora biti 0, a
     d je jednako poslednjem validnom indeksu u nizu. Funkcija niz
     podeli na dve polovine, levu i desnu, koje zatim rekurzivno
     sortira. Od ova dva sortirana podniza, sortiran niz se dobija
     ucesljavanjem, tj. istovremenim prolaskom kroz oba niza i izborom
     trenutnog manjeg elementa koji se smesta u pomocni niz. Na kraju
64
     algoritma, sortirani elementi su u pomocnom nizu, koji se kopira u
```

```
originalni niz. */
void merge_sort(int a[], int l, int d);

/* Quick sort: Funkcija sortira deo niza brojeva a izmedju pozicija
l i d. Njena ideja sortiranja je izbor jednog elementa niza, koji
se naziva pivot, i koji se dovodi na svoje mesto. Posle ovog
koraka, svi elementi levo od njega bice manji, a svi desno bice
veci od njega. Kako je pivot doveden na svoje mesto, da bi niz
bio kompletno sortiran, potrebno je sortirati elemente levo
(manje) od njega, i elemente desno (vece). Kako su dimenzije ova
dva podniza manje od dimenzije pocetnog niza koji je trebalo
sortirati, ovaj deo moze se uraditi rekurzivno. */
void quick_sort(int a[], int l, int d);

#endif
```

#### sort.c

```
#include "sort.h"
  #define MAX 1000000
  void selection_sort(int a[], int n)
    int i, j;
    int min;
    int pom;
    /* U svakoj iteraciji ove petlje pronalazi se najmanji element
       medju elementima a[i], a[i+1],...,a[n-1], i postavlja se na
12
       poziciju i, dok se element na pozciji i premesta na poziciju
       min, na kojoj se nalazio najmanji od navedenih elemenata. */
14
    for (i = 0; i < n - 1; i++) {
      /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
         najmanji od elemenata a[i],...,a[n-1]. */
18
      min = i;
      for (j = i + 1; j < n; j++)
20
        if (a[j] < a[min])
          min = j;
      /* Zamena elemenata na pozicijama (i) i min. Ovo se radi samo
         ako su (i) i min razliciti, inace je nepotrebno. */
24
      if (min != i) {
        pom = a[i];
        a[i] = a[min];
        a[min] = pom;
28
    }
30
  }
  void insertion_sort(int a[], int n)
```

```
34 {
    int i, j;
36
    /* Na pocetku iteracije pretpostavlja se da je niz a[0],...,a[i-1]
       sortiran */
38
    for (i = 1; i < n; i++) {
      /* U ovoj petlji se redom potiskuje element a[i] ulevo koliko je
40
         potrebno, dok ne zauzme pravo mesto, tako da niz a[0],...a[i]
         bude sortiran. Indeks j je trenutna pozicija na kojoj se
         element koji se umece nalazi. Petlja se zavrsava ili kada
         element dodje do levog kraja (j==0) ili kada se naidje na
44
         element a[j-1] koji je manji od a[j]. */
      int temp = a[i];
46
      for (j = i; j > 0 \&\& temp < a[j - 1]; j--)
        a[j] = a[j - 1];
48
      a[j] = temp;
  }
  void bubble_sort(int a[], int n)
  {
54
    int i, j;
    int ind;
    for (i = n, ind = 1; i > 1 \&\& ind; i--)
      /* Poput "mehurica" potiskuje se najveci element medju
         elementima od a[0] do a[i-1] na poziciju i-1 uporedjujuci
         susedne elemente niza i potiskujuci veci u desno */
      for (j = 0, ind = 0; j < i - 1; j++)
        if (a[j] > a[j + 1]) {
          int temp = a[j];
          a[j] = a[j + 1];
          a[j + 1] = temp;
           /* Promenljiva ind registruje da je bilo premestanja. Samo u
             tom slucaju ima smisla ici na sledecu iteraciju, jer ako
             nije bilo premestanja, znaci da su svi elementi vec u
             dobrom poretku, pa nema potrebe prelaziti na kraci prefiks
             niza. Algoritam moze biti i bez ovoga, sortiranje bi bilo
              ispravno, ali manje efikasano, jer bi se cesto nepotrebno
              vrsila mnoga uporedjivanja, kada je vec jasno da je
              sortiranje zavrseno. */
           ind = 1;
        }
76
  void shell_sort(int a[], int n)
80
    int h = n / 2, i, j;
    while (h > 0) {
82
      /* Insertion sort sa korakom h */
      for (i = h; i < n; i++) \{
        int temp = a[i];
```

```
j = i;
86
         while (j \ge h \&\& a[j - h] \ge temp) {
           a[j] = a[j - h];
88
           j -= h;
90
         a[j] = temp;
92
       h = h / 2;
   }
96
   void merge_sort(int a[], int 1, int d)
  {
98
     int s;
     static int b[MAX];
                                    /* Pomocni niz */
100
     int i, j, k;
     /* Izlaz iz rekurzije */
     if (1 >= d)
104
      return;
106
     /* Odredjivanje sredisnjeg indeksa */
     s = (1 + d) / 2;
108
     /* Rekurzivni pozivi */
     merge_sort(a, 1, s);
     merge_sort(a, s + 1, d);
112
     /* Inicijalizacija indeksa. Indeks i prolazi krozi levu polovinu
114
        niza, dok indeks j prolazi kroz desnu polovinu niza. Indeks k
        prolazi kroz pomocni niz b[] */
     i = 1;
     j = s + 1;
118
     k = 0;
120
     /* "Ucesljavanje" koriscenjem pomocnog niza b[] */
     while (i <= s && j <= d) {
       if (a[i] < a[j])
         b[k++] = a[i++];
124
       else
         b[k++] = a[j++];
126
128
     /* U slucaju da se prethodna petlja zavrsila izlaskom promenljive
        j iz dopustenog opsega u pomocni niz se prepisuje ostatak leve
130
        polovine niza */
     while (i <= s)
       b[k++] = a[i++];
134
     /* U slucaju da se prethodna petlja zavrsila izlaskom promenljive
        i iz dopustenog opsega u pomocni niz se prepisuje ostatak desne
136
        polovine niza */
```

```
while (j \le d)
138
       b[k++] = a[j++];
140
     /* Prepisuje se "ucesljani" niz u originalni niz */
     for (k = 0, i = 1; i \le d; i++, k++)
142
       a[i] = b[k];
  }
144
   /* Pomocna funkcija koja menja mesto i-tom i j-tom elementu niza a */
146
   void swap(int a[], int i, int j)
148
     int tmp = a[i];
     a[i] = a[j];
     a[j] = tmp;
  }
   void quick_sort(int a[], int 1, int d)
154
     int i, pivot_pozicija;
     /* Izlaz iz rekurzije -- prazan niz */
158
     if (1 >= d)
       return;
160
     /* Particionisanje niza. Svi elementi na pozicijama levo od
        pivot_pozicija (izuzev same pozicije 1) su strogo manji od
        pivota. Kada se pronadje neki element manji od pivota, uvecava
164
        se promenljiva pivot_pozicija i na tu poziciju se premesta
        nadjeni element. Na kraju ce pivot_pozicija zaista biti
        pozicija na koju treba smestiti pivot, jer ce svi elementi levo
        od te pozicije biti manji a desno biti veci ili jednaki od
168
        pivota. */
     pivot_pozicija = 1;
     for (i = 1 + 1; i <= d; i++)
172
       if (a[i] < a[l])
         swap(a, ++pivot_pozicija, i);
174
     /* Postavljanje pivota na svoje mesto */
     swap(a, 1, pivot_pozicija);
     /* Rekurzivno sortiranje elementa manjih od pivota */
     quick_sort(a, 1, pivot_pozicija - 1);
     /* Rekurzivno sortiranje elementa vecih od pivota */
     quick_sort(a, pivot_pozicija + 1, d);
   3
182
```

#### main.c

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

```
4 #include "sort.h"
 /* Maksimalna duzina niza */
  #define MAX 1000000
  int main(int argc, char *argv[])
10 1
    /*************************
      tip_sortiranja == 0 => selectionsort, (podrazumevano)
      tip_sortiranja == 1 => insertionsort, -i opcija komandne linije
      tip_sortiranja == 2 => bubblesort, -b opcija komandne linije
tip_sortiranja == 3 => shellsort, -s opcija komandne linije
tip_sortiranja == 4 => mergesort, -m opcija komandne linije
14
      tip_sortiranja == 5 => quicksort,
                                         -q opcija komandne linije
    *********************
18
    int tip_sortiranja = 0;
    20
      tip_niza == 0 => slucajno generisani nizovi, (podrazumevano)
      tip_niza == 1 => rastuce sortirani nizovi, -r opcija
      tip_niza == 2 => opadajuce soritrani nizovi, -o opcija
    24
    int tip_niza = 0;
26
    /* Dimenzija niza koji se sortira */
   int dimenzija;
28
   int i:
   int niz[MAX];
30
    /* Provera argumenata komandne linije */
   if (argc < 2) {
     fprintf(stderr, "Greska: Program zahteva bar 2 ");
34
     fprintf(stderr, "argumenta komandne linije!\n");
     exit(EXIT_FAILURE);
36
38
    /* Ocitavaju se opcije i argumenati komandne linije */
   for (i = 1; i < argc; i++) {
40
     /* Ako je u pitanju opcija... */
     if (argv[i][0] == '-') {
42
       switch (argv[i][1]) {
       case 'i':
44
         tip_sortiranja = 1;
46
         break:
       case 'b':
         tip_sortiranja = 2;
48
         break;
       case 's':
50
         tip_sortiranja = 3;
         break;
       case 'm':
         tip_sortiranja = 4;
54
         break:
```

```
case 'q':
56
          tip_sortiranja = 5;
          break:
58
         case 'r':
          tip_niza = 1;
60
          break;
         case 'o':
          tip_niza = 2;
          break:
         default:
          fprintf(stderr, "Greska: Pogresna opcija -%c\n", argv[i][1]);
           exit(EXIT_SUCCESS);
          break:
68
        }
       /* Ako je u pitanju argument, onda je to duzina niza koji treba
         da se sortira */
       else {
         dimenzija = atoi(argv[i]);
74
         if (dimenzija <= 0 || dimenzija > MAX) {
          fprintf(stderr, "Greska: Dimenzija niza neodgovarajuca!\n");
76
           exit(EXIT_FAILURE);
        }
      }
80
     /* Elementi niza se odredjuju slucajno, ali vodeci racuna o tipu
82
       niza dobijenom iz komandne linije. srand() funkcija obezbedjuje
       novi seed za pozivanje rand funkcije, i kako generisani niz ne
84
       bi uvek bio isti seed je postavljen na tekuce vreme u sekundama
       od Nove godine 1970. rand()%100 daje brojeve izmedju 0 i 99 */
86
     srand(time(NULL));
     if (tip_niza == 0)
88
      for (i = 0; i < dimenzija; i++)
        niz[i] = rand();
90
     else if (tip_niza == 1)
      for (i = 0; i < dimenzija; i++)</pre>
92
        niz[i] = i == 0 ? rand() % 100 : niz[i - 1] + rand() % 100;
94
     else
      for (i = 0; i < dimenzija; i++)</pre>
         niz[i] = i == 0 ? rand() % 100 : niz[i - 1] - rand() % 100;
96
     /* Ispisuju se elemenati niza */
98
     Ovaj deo je iskomentarisan jer sledeci ispis ne treba da se nadje
100
      na standardnom izlazu. Njegova svrha je samo bila provera da li
      je niz generisan u skladu sa opcijama komandne linije.
      printf("Niz koji sortiramo je:\n");
104
      for (i = 0; i < dimenzija; i++)</pre>
        printf("%d\n", niz[i]);
106
                                      ************
```

```
108
    /* Sortira se niz na odgovarajuci nacin */
    if (tip_sortiranja == 0)
     selection_sort(niz, dimenzija);
    else if (tip_sortiranja == 1)
     insertion_sort(niz, dimenzija);
    else if (tip_sortiranja == 2)
114
     bubble_sort(niz, dimenzija);
    else if (tip_sortiranja == 3)
     shell_sort(niz, dimenzija);
    else if (tip_sortiranja == 4)
118
     merge_sort(niz, 0, dimenzija - 1);
    else
      quick_sort(niz, 0, dimenzija - 1);
    /* Ispisuju se elemenati niza */
    124
      Ovaj deo je iskomentarisan jer vreme potrebno za njegovo
      izvrsavanje ne bi trebalo da bude ukljuceno u vreme izmereno
      programom time. Takodje, kako je svrha ovog programa da prikaze
      vremena razlicitih algoritama sortiranja, dimenzije nizova ce
128
      biti, verovatno, ogromne, pa nema smisla imati na izlazu nizove
      od toliko elemenata. Ovaj deo je koriscen u razvoju programa
130
      zarad testiranja korektnosti.
      printf("Sortiran niz je:\n");
      for (i = 0; i < dimenzija; i++)</pre>
134
       printf("%d\n", niz[i]);
      136
    exit(EXIT_SUCCESS);
138
```

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

#define MAX_DIM 128

/* Funkcija za sortiranje niza karaktera */
void selectionSort(char s[])
{
   int i, j, min;
   char pom;
   for (i = 0; s[i] != '\0'; i++) {
      min = i;
      for (j = i + 1; s[j] != '\0'; j++)
      if (s[j] < s[min])
            min = j;
   if (min != i) {</pre>
```

```
pom = s[i];
         s[i] = s[min];
         s[min] = pom;
20
22 }
  /* Funkcija vraca 1 ako su argumenti anagrami, a 0 inace. */
  int anagrami(char s[], char t[])
26
    int i;
28
    /* Ako dve niske imaju razlicit broj karaktera onda one nisu
       anagrami */
30
    if (strlen(s) != strlen(t))
      return 0;
    /* Sortiraju se karakteri */
34
    selectionSort(s);
    selectionSort(t);
36
    /* Dve sortirane niske su anagrami ako i samo ako su jednake */
    for (i = 0; s[i] != '\0'; i++)
      if (s[i] != t[i])
40
        return 0;
    return 1;
42
44
  int main()
  {
46
    char s[MAX_DIM], t[MAX_DIM];
48
    /* Ucitavaju se niske sa ulaza */
    printf("Unesite prvu nisku: ");
50
    scanf("%s", s);
    printf("Unesite drugu nisku: ");
    scanf("%s", t);
54
     /* Poziv funkcije */
    if (anagrami(s, t))
56
      printf("jesu\n");
    else
58
      printf("nisu\n");
60
    return 0;
62 }
```

NAPOMENA: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka 3.12.

```
#include <stdio.h>
  #include "sort.h"
  #define MAX 256
  /* Funkcija koja pronalazi najmanje rastojanje izmedju dva broja u
     sortiranom nizu celih brojeva */
  int najmanje_rastojanje(int a[], int n)
9
    int i, min;
    /* Postavlja se najmanje rastojanje na razliku prvog i drugog
11
       elementa niza */
    min = a[1] - a[0];
13
    /* Za sve ostale susedne elemente proverava se njigova razlika */
    for (i = 2; i < n; i++)
      if (a[i] - a[i - 1] < min)
       min = a[i] - a[i - 1];
    return min;
 | }
  int main()
21
    int i, a[MAX];
    /* Ucitavaju se elementi niza sve do kraja ulaza */
    while (scanf("%d", &a[i]) != EOF)
      i++;
    /* Za sortiranje niza moze se koristiti bilo koja od funkcija
       sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
       se selection sort. */
    selection_sort(a, i);
    /* Ispisuje se rezultat */
    printf("%d\n", najmanje_rastojanje(a, i));
    return 0;
39
```

# Rešenje 3.15

Napomena: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka 3.12.

```
#include <stdio.h>
#include "sort.h"

#define MAX_DIM 256
```

```
/* Funkcija za odredjivanje onog elementa sortiranog niza koji se
     najvise puta pojavio u tom nizu */
  int najvise_puta(int a[], int n)
  {
9
    int i, j, br_pojava, i_max_pojava = -1, max_br_pojava = -1;
    /* Za i-ti element izracunava se koliko puta se pojavio u nizu */
    for (i = 0; i < n; i = j) {
      br_pojava = 1;
13
      for (j = i + 1; j < n \&\& a[i] == a[j]; j++)
        br_pojava++;
      /* Ispituje se da li se do tog trenutka i-ti element pojavio
         najvise puta u nizu */
      if (br_pojava > max_br_pojava) {
        max_br_pojava = br_pojava;
19
        i_max_pojava = i;
    /* Vraca se element koji se najvise puta pojavio u nizu */
    return a[i_max_pojava];
25
  int main()
    int a[MAX_DIM], i;
29
    /* Ucitavaju se elemenati niza sve do kraja ulaza */
    while (scanf("%d", &a[i]) != EOF)
      i++;
    /* Za sortiranje niza moze se koristiti bilo koja od funkcija
       sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
37
       se merge sort. */
    merge_sort(a, 0, i - 1);
39
    /* Odredjuje se broj koji se najvise puta pojavio u nizu */
41
    printf("%d\n", najvise_puta(a, i));
43
    return 0;
  }
45
```

Napomena: Rešenje koristi biblioteku za sortiranje celih brojeva iz zadatka 3.12.

```
#include <stdio.h>
#include "sort.h"

#define MAX_DIM 256
```

```
/* Funkcija za binarnu pretragu niza vraca 1 ako se element x nalazi
     u nizu, a O inace. Pretpostavlja se da je niz sortiran u rastucem
     poretku */
9 int binarna_pretraga(int a[], int n, int x)
    int levi = 0, desni = n - 1, srednji;
    while (levi <= desni) {
      srednji = (levi + desni) / 2;
      if (a[srednji] == x)
        return 1;
      else if (a[srednji] > x)
17
        desni = srednji - 1;
      else if (a[srednji] < x)</pre>
19
        levi = srednji + 1;
    }
    return 0;
23 }
25 int main()
    int a[MAX_DIM], n = 0, zbir, i;
    /* Ucitava se trazeni zbir */
    printf("Unesite trazeni zbir: ");
    scanf("%d", &zbir);
    /* Ucitavaju se elementi niza sve do kraja ulaza */
33
    i = 0;
    printf("Unesite elemente niza: ");
35
    while (scanf("%d", &a[i]) != EOF)
     i++:
37
    n = i;
    /* Za sortiranje niza moze se koristiti bilo koja od funkcija
       sortiranja iz sort.h. Ilustracije radi, u ovom zadatku koristi
41
       se quick sort. */
    quick_sort(a, 0, n - 1);
43
    for (i = 0; i < n; i++)
45
      /* Za i-ti element niza binarno se pretrazuje da li se u ostatku
         niza nalazi element koji sabran sa njim ima ucitanu vrednost
47
         zbira */
      if (binarna_pretraga(a + i + 1, n - i - 1, zbir - a[i])) {
49
        printf("da\n");
        return 0;
    printf("ne\n");
    return 0;
```

```
#include <stdio.h>
  #define MAX_DIM 256
  /* Funkcija objedinjuje nizove niz1 i niz2 dimenzija dim1 i dim2, a
     rezultat cuva u nizu dim3 za koji je rezervisano dim3 elemenata */
  int merge(int *niz1, int dim1, int *niz2, int dim2, int *niz3,
            int dim3)
    int i = 0, j = 0, k = 0;
    /* U slucaju da je dimenzija treceg niza manja od neophodne,
       funkcija vraca -1 */
    if (\dim 3 < \dim 1 + \dim 2)
      return -1;
    /* Vrsi se ucesljavanje nizova sve dok se ne dodje do kraja jednog
       od njih */
    while (i < dim1 && j < dim2) {
      if (niz1[i] < niz2[j])</pre>
        niz3[k++] = niz1[i++];
20
      else
        niz3[k++] = niz2[j++];
    /* Ostatak prvog niza prepisuje se u treci */
    while (i < dim1)
      niz3[k++] = niz1[i++];
    /* Ostatak drugog niza prepisuje se u treci */
    while (j < dim2)
      niz3[k++] = niz2[j++];
    return dim1 + dim2;
  int main()
    int niz1[MAX_DIM], niz2[MAX_DIM], niz3[2 * MAX_DIM];
    int i = 0, j = 0, k, dim3;
    /* Ucitavaju se nizovi sa ulaza sve dok se ne unese nula.
38
       Pretpostavka je da na ulazu nema vise od MAX_DIM elemenata */
    printf("Unesite elemente prvog niza: ");
    while (1) {
      scanf("%d", &niz1[i]);
      if (niz1[i] == 0)
        break;
      i++;
46
    printf("Unesite elemente drugog niza: ");
    while (1) {
48
      scanf("%d", &niz2[j]);
      if (niz2[j] == 0)
```

```
break;
   j++;
}

/* Poziv trazene funkcije */
dim3 = merge(niz1, i, niz2, j, niz3, 2 * MAX_DIM);

/* Ispis niza */
for (k = 0; k < dim3; k++)
   printf("%d ", niz3[k]);
   printf("\n");

return 0;

4
}</pre>
```

```
#include <stdio.h>
2 #include <stdlib.h>
  #include <string.h>
  int main(int argc, char *argv[])
6 {
    FILE *fin1 = NULL, *fin2 = NULL;
   FILE *fout = NULL;
    char ime1[11], ime2[11];
   char prezime1[16], prezime2[16];
10
    int kraj1 = 0, kraj2 = 0;
12
    /* Ako nema dovoljno arguemenata komandne linije */
14
    if (argc < 3) {
      fprintf(stderr,
              "Greska: Program se poziva sa %s datoteka1 datoteka2\n",
16
              argv[0]);
      exit(EXIT_FAILURE);
18
    }
20
    /* Otvara se datoteka zadata prvim argumentom komandne linije */
    fin1 = fopen(argv[1], "r");
    if (fin1 == NULL) {
24
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s\n",
              argv[1]);
      exit(EXIT_FAILURE);
26
    }
28
    /* Otvara se datoteka zadata drugim argumentom komandne linije */
    fin2 = fopen(argv[2], "r");
30
    if (fin2 == NULL) {
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s\n",
32
              argv[2]);
34
      exit(EXIT_FAILURE);
```

```
/* Otvara se datoteka za upis rezultata */
    fout = fopen("ceo-tok.txt", "w");
38
    if (fout == NULL) {
      fprintf(stderr,
40
               "Greska: Neuspesno otvaranje datoteke ceo-tok.txt\n");
      exit(EXIT_FAILURE);
42
44
    /* Cita se prvi student iz prve datoteke */
    if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
46
      kraj1 = 1;
48
    /* Cita se prvi student iz druge datoteke */
    if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
      kraj2 = 1;
    /* Sve dok nije dostignut kraj neke datoteke */
    while (!kraj1 && !kraj2) {
54
      int tmp = strcmp(ime1, ime2);
      if (tmp < 0 \mid | (tmp == 0 \&\& strcmp(prezime1, prezime2) < 0)) {
56
        /* Ime i prezime iz prve datoteke je leksikografski ranije, i
           biva upisano u izlaznu datoteku */
58
        fprintf(fout, "%s %s\n", ime1, prezime1);
        /* Cita se naredni student iz prve datoteke */
        if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
62
          kraj1 = 1;
      } else {
        /* Ime i prezime iz druge datoteke je leksikografski ranije, i
64
           biva upisano u izlaznu datoteku */
        fprintf(fout, "%s %s\n", ime2, prezime2);
        /* Cita se naredni student iz druge datoteke */
        if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
          kraj2 = 1;
      }
    }
    /* Ako se iz prethodne petlje izaslo zato sto je dostignut kraj
       druge datoteke, onda ima jos studenata u prvoj datoteci, koje
74
       treba prepisati u izlaznu, redom, jer su vec sortirani po
       imenu. */
76
    while (!kraj1) {
      fprintf(fout, "%s %s\n", ime1, prezime1);
      if (fscanf(fin1, "%s%s", ime1, prezime1) == EOF)
        kraj1 = 1;
80
82
    /* Ako se iz prve petlje izaslo zato sto je dostignut kraj prve
       datoteke, onda ima jos studenata u drugoj datoteci, koje treba
84
       prepisati u izlaznu, redom, jer su vec sortirani po imenu. */
    while (!kraj2) {
```

```
fprintf(fout, "%s %s\n", ime2, prezime2);
if (fscanf(fin2, "%s%s", ime2, prezime2) == EOF)
    kraj2 = 1;
}

/* Zatvaraju se datoteke */
fclose(fin1);
fclose(fin2);
fclose(fout);

exit(EXIT_SUCCESS);
```

```
#include <stdio.h>
  #include <string.h>
3 #include <math.h>
  #include <stdlib.h>
  #define MAX_BR_TACAKA 128
  /* Struktura koja reprezentuje koordinate tacke */
9 typedef struct Tacka {
   int x;
   int y;
  } Tacka;
  /* Funkcija racuna rastojanje zadate tacke od koordinatnog pocetka */
15 float rastojanje (Tacka A)
    return sqrt(A.x * A.x + A.y * A.y);
  }
19
  /* Funkcija koja sortira niz tacaka po rastojanju od koordinatnog
    pocetka */
  void sortiraj_po_rastojanju(Tacka t[], int n)
23 {
    int min, i, j;
25
   Tacka tmp;
27
    for (i = 0; i < n - 1; i++) {
      min = i;
      for (j = i + 1; j < n; j++) {
29
        if (rastojanje(t[j]) < rastojanje(t[min])) {</pre>
          min = j;
        }
      }
33
      if (min != i) {
        tmp = t[i];
35
        t[i] = t[min];
```

```
t[min] = tmp;
39
  }
41
  /* Funkcija koja sortira niz tacaka po vrednosti x koordinate */
  void sortiraj_po_x(Tacka t[], int n)
43
    int min, i, j;
45
    Tacka tmp;
47
    for (i = 0; i < n - 1; i++) {
      min = i;
49
      for (j = i + 1; j < n; j++) {
        if (abs(t[j].x) < abs(t[min].x)) {
51
          min = j;
53
      }
      if (min != i) {
        tmp = t[i];
        t[i] = t[min];
        t[min] = tmp;
59
  }
61
  /* Funkcija koja sortira niz tacaka po vrednosti y koordinate */
  void sortiraj_po_y(Tacka t[], int n)
65
    int min, i, j;
67
    Tacka tmp;
    for (i = 0; i < n - 1; i++) {
69
      min = i;
      for (j = i + 1; j < n; j++) {
        if (abs(t[j].y) < abs(t[min].y)) {
          min = j;
73
        }
      }
      if (min != i) {
        tmp = t[i];
        t[i] = t[min];
        t[min] = tmp;
79
81
83
  int main(int argc, char *argv[])
85
    FILE *ulaz;
    FILE *izlaz;
    Tacka tacke[MAX_BR_TACAKA];
```

```
89
     int i, n;
     /* Proverava se broj argumenata komandne linije: ocekuje se ime
91
        izvrsnog programa, opcija, ime ulazne datoteke i ime izlazne
        datoteke, tj. 4 argumenta */
93
     if (argc != 4) {
      fprintf(stderr,
95
               "Greska: Program se poziva sa %s opcija ulaz izlaz\n",
               argv[0]);
       exit(EXIT_FAILURE);
99
     /* Otvara se datoteka u kojoj su zadate tacke */
     ulaz = fopen(argv[2], "r");
     if (ulaz == NULL) {
       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
               argv[2]);
       exit(EXIT_FAILURE);
     /* Otvara se datoteka u koju treba upisati rezultat */
     izlaz = fopen(argv[3], "w");
     if (izlaz == NULL) {
       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s!\n",
               argv[3]);
113
       exit(EXIT_FAILURE);
     }
     /* Sve dok se ne stigne do kraja ulazne datoteke, ucitavaju se
        koordinate tacaka i smestaju na odgovarajuce pozicije odredjene
        brojacem i. */
119
     i = 0;
     while (fscanf(ulaz, "%d %d", &tacke[i].x, &tacke[i].y) != EOF) {
     /* Ukupan broj procitanih tacaka */
     n = i;
     /* Analizira se prosledjena opcija. Moguce vrednosti za argv[1] su
        "-x" ili "-y" ili "-o", pa je argv[1][0] sigurno crtica
        (karakter -), a karakter argv[1][1] odredjuje kriterijum
        sortiranja */
     switch (argv[1][1]) {
     case 'x':
133
       /* Sortira se po vrednosti x koordinate */
      sortiraj_po_x(tacke, n);
      break;
     case 'y':
      /* Sortira se po vrednosti y koordinate */
       sortiraj_po_y(tacke, n);
139
       break;
```

```
case 'o':
141
       /* Sortira se po udaljenosti od koorinatnog pocetka */
       sortiraj_po_rastojanju(tacke, n);
143
       break;
145
     /* Upisuje se dobijeni niz u izlaznu datoteku */
147
     for (i = 0; i < n; i++) {
      fprintf(izlaz, "%d %d\n", tacke[i].x, tacke[i].y);
     /* Zatvaraju se otvorene datoteke */
     fclose(ulaz);
     fclose(izlaz);
     exit(EXIT_SUCCESS);
  }
157
```

```
#include <stdio.h>
  #include <string.h>
  #include <stdlib.h>
  #define MAX 1000
6 #define MAX_DUZINA 16
  /* Struktura koja reprezentuje jednog gradjanina */
  typedef struct gr {
    char ime[MAX_DUZINA];
    char prezime[MAX_DUZINA];
12 } Gradjanin;
14 /* Funkcija sortira niz gradjana rastuce po imenima */
  void sort_ime(Gradjanin a[], int n)
16
    int i, j;
18
    int min;
    Gradjanin pom;
    for (i = 0; i < n - 1; i++) {
      /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
22
         najmanji od elemenata a[i].ime,...,a[n-1].ime. */
      min = i;
24
      for (j = i + 1; j < n; j++)
        if (strcmp(a[j].ime, a[min].ime) < 0)</pre>
26
          min = j;
      /* Zamena elemenata na pozicijama (i) i min. Ovo se radi samo
         ako su (i) i min razliciti, inace je nepotrebno. */
      if (min != i) {
30
        pom = a[i];
```

```
a[i] = a[min];
32
        a[min] = pom;
34
    }
  }
36
38 /* Funkcija sortira niz gradjana rastuce po prezimenima */
  void sort_prezime(Gradjanin a[], int n)
40 {
    int i, j;
    int min;
42
    Gradjanin pom;
44
    for (i = 0; i < n - 1; i++) {
      /* Unutrasnja petlja pronalazi poziciju min, na kojoj se nalazi
46
         najmanji od elemenata a[i].prezime,...,a[n-1].prezime. */
      min = i;
48
      for (j = i + 1; j < n; j++)
        if (strcmp(a[j].prezime, a[min].prezime) < 0)</pre>
          min = j;
      /* Zamena elemenata na pozicijama (i) i min. Ovo se radi samo
         ako su (i) i min razliciti, inace je nepotrebno. */
      if (min != i) {
        pom = a[i];
        a[i] = a[min];
56
        a[min] = pom;
      }
58
    }
  }
60
62 /* Pretraga niza gradjana */
  int linearna_pretraga(Gradjanin a[], int n, Gradjanin * x)
64 {
    int i;
    for (i = 0; i < n; i++)
      if (strcmp(a[i].ime, x->ime) == 0
          && strcmp(a[i].prezime, x->prezime) == 0)
68
        return i;
    return -1;
70
72
74 int main()
    Gradjanin spisak1[MAX], spisak2[MAX];
76
    int isti_rbr = 0;
    int i, n;
78
    FILE *fp = NULL;
80
    /* Otvara se datoteka */
    if ((fp = fopen("biracki-spisak.txt", "r")) == NULL) {
82
      fprintf(stderr,
```

```
"Greska: Neupesno otvaranje datoteke za citanje.\n");
84
      exit(EXIT FAILURE);
86
    /* Cita se sadrzaj */
88
    for (i = 0;
        fscanf(fp, "%s %s", spisak1[i].ime,
90
               spisak1[i].prezime) != EOF; i++)
      spisak2[i] = spisak1[i];
92
    n = i;
94
    /* Zatvara se datoteka */
    fclose(fp);
96
    sort_ime(spisak1, n);
98
    /************************
100
      Ovaj deo je iskomentarisan jer se u zadatku ne trazi ispis
      sortiranih nizova. Koriscen je samo u fazi testiranja programa.
      printf("Biracki spisak [uredjen prema imenima]:\n");
104
      for(i=0; i<n; i++)
       printf(" %d. %s %s\n",i,spisak1[i].ime, spisak1[i].prezime);
106
108
    sort_prezime(spisak2, n);
    Ovaj deo je iskomentarisan jer se u zadatku ne trazi ispis
      sortiranih nizova. Koriscen je samo u fazi testiranja programa.
114
      printf("Biracki spisak [uredjen prema prezimenima]:\n");
      for(i=0; i<n; i++)
       printf(" %d. %s %s\n",i,spisak2[i].ime, spisak2[i].prezime);
118
120
    /* Linearno se pretrazuju nizovi */
    for (i = 0; i < n; i++)
      if (i == linearna_pretraga(spisak2, n, &spisak1[i]))
        isti_rbr++;
124
    /* Alternativno (efikasnije) resenje */
    /***********************
126
      for(i=0; i<n;i++)
        if( strcmp(spisak2[i].ime, spisak1[i].ime) == 0 &&
128
           strcmp(spisak1[i].prezime, spisak2[i].prezime)==0)
130
      isti_rbr++;
    /* Ispisuje se rezultat */
    printf("%d\n", isti_rbr);
134
```

```
exit(EXIT_SUCCESS);
}
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
  #include <ctype.h>
  #define MAX_BR_RECI 128
7 #define MAX_DUZINA_RECI 32
9 /* Funkcija koja izracunava broj suglasnika u reci */
  int broj_suglasnika(char s[])
11 {
    char c;
   int i;
    int suglasnici = 0;
    /* Prolaz karakter po karakter kroz zadatu nisku */
    for (i = 0; s[i]; i++) {
      /* Ako je u pitanju slovo, konvertuje se u veliko da bi bio
17
         pokriven slucaj i malih i velikih suglasnika. */
      if (isalpha(s[i])) {
        c = toupper(s[i]);
        /* Ukoliko slovo nije samoglasnik uvecava se brojac. */
        if (c != 'A' && c != 'E' && c != 'I' && c != 'O' && c != 'U')
          suglasnici++;
      }
    }
    /* Vraca se izracunata vrednost */
    return suglasnici;
29
  /* Funkcija koja sortira reci po zadatom kriterijumu. Informacija o
31
     duzini reci se mora proslediti zbog pravilnog upravljanja
     memorijom */
33 void sortiraj_reci(char reci[][MAX_DUZINA_RECI], int n)
    int min, i, j, broj_suglasnika_j, broj_suglasnika_min,
35
        duzina_j, duzina_min;
    char tmp[MAX_DUZINA_RECI];
    for (i = 0; i < n - 1; i++) {
      min = i;
39
      for (j = i; j < n; j++) {
        /* Prvo se uporedjuje broj suglasnika */
41
        broj_suglasnika_j = broj_suglasnika(reci[j]);
        broj_suglasnika_min = broj_suglasnika(reci[min]);
43
        if (broj_suglasnika_j < broj_suglasnika_min)</pre>
          min = j;
45
        else if (broj_suglasnika_j == broj_suglasnika_min) {
```

```
47
           /* Zatim, recima koje imaju isti broj suglasnika uporedjuju
              se duzine */
           duzina_j = strlen(reci[j]);
49
           duzina_min = strlen(reci[min]);
           if (duzina_j < duzina_min)</pre>
            min = j;
53
           else {
             /* Ako reci imaju i isti broj suglasnika i iste duzine,
                uporedjuju se leksikografski */
             if (duzina_j == duzina_min
                 && strcmp(reci[j], reci[min]) < 0)
               min = j;
          }
        }
      }
      if (min != i) {
63
        strcpy(tmp, reci[min]);
        strcpy(reci[min], reci[i]);
         strcpy(reci[i], tmp);
67
  }
69
  int main()
71
    FILE *ulaz;
73
    int i = 0, n;
    /* Niz u koji ce biti smestane reci. Prvi broj oznacava broj reci,
       a drugi maksimalnu duzinu pojedinacne reci */
    char reci[MAX_BR_RECI][MAX_DUZINA_RECI];
79
    /* Otvara se datoteka niske.txt za citanje */
    ulaz = fopen("niske.txt", "r");
81
    if (ulaz == NULL) {
83
      fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke niske.txt!\n");
      exit(EXIT_FAILURE);
85
87
    /* Sve dok se moze procitati sledeca rec */
    while (fscanf(ulaz, "%s", reci[i]) != EOF) {
89
      /* Proverava se da li ucitan maksimalan broj reci, i ako jeste,
          prekida se ucitavanje */
91
      if (i == MAX_BR_RECI)
93
        break:
      /* Priprema brojaca za narednu iteraciju */
95
      i++;
97
    /* n je duzina niza reci i predstavlja poslednju vrednost
```

```
koriscenog brojaca */
n = i;
/* Poziva se funkcija za sortiranje reci */
sortiraj_reci(reci, n);

/* Ispis sortiranog niza reci */
for (i = 0; i < n; i++) {
    printf("%s ", reci[i]);
}
printf("\n");

/* Zatvara se datoteka */
fclose(ulaz);

exit(EXIT_SUCCESS);
}</pre>
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #define MAX_ARTIKALA 100000
  /* Struktura koja predstavlja jedan artikal */
8 typedef struct art {
   long kod;
  char naziv[20];
   char proizvodjac[20];
12
   float cena;
  } Artikal;
14
  /* Funkcija koja u nizu artikala binarnom pretragom nalazi onaj sa
    trazenim bar kodom */
  int binarna_pretraga(Artikal a[], int n, long x)
18 {
    int levi = 0;
    int desni = n - 1;
20
    /* Dokle god je indeks levi levo od indeksa desni */
    while (levi <= desni) {
      /* Racuna se sredisnji indeks */
24
      int srednji = (levi + desni) / 2;
      /* Ako je sredisnji element veci od trazenog, tada se trazeni
26
         mora nalaziti u levoj polovini niza */
      if (x < a[srednji].kod)</pre>
28
        desni = srednji - 1;
      /* Ako je sredisnji element manji od trazenog, tada se trazeni
30
         mora nalaziti u desnoj polovini niza */
      else if (x > a[srednji].kod)
32
```

```
levi = srednji + 1;
      else
34
         /* Ako je sredisnji element jednak trazenom, tada je artikal
           sa bar kodom x pronadjen na poziciji srednji */
36
        return srednji;
38
    /* Ako nije pronadjen artikal za trazenim bar kodom, vraca se -1 */
    return -1;
40
42
  /* Funkcija koja sortira niz artikala po bar kodovima rastuce */
44 void selection_sort(Artikal a[], int n)
    int i, j;
46
    int min;
    Artikal pom;
48
    for (i = 0; i < n - 1; i++) {
50
      min = i;
      for (j = i + 1; j < n; j++)
        if (a[j].kod < a[min].kod)</pre>
          min = j;
54
      if (min != i) {
        pom = a[i];
56
        a[i] = a[min];
        a[min] = pom;
58
60
62
  int main()
64
    Artikal asortiman[MAX_ARTIKALA];
    long kod;
66
    int i, n;
    float racun;
68
    FILE *fp = NULL;
    /* Otvara se datoteka */
    if ((fp = fopen("artikli.txt", "r")) == NULL) {
72
      fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke artikli.txt.\n");
74
      exit(EXIT_FAILURE);
    }
76
    /* Ucitavaju se artikali */
    i = 0;
    while (fscanf(fp, "%ld %s %s %f", &asortiman[i].kod,
80
                   asortiman[i].naziv, asortiman[i].proizvodjac,
                   &asortiman[i].cena) == 4)
82
      i++;
84
```

```
/* Zatvara se datoteka */
    fclose(fp);
86
    n = i;
88
     /* Sortira se celokupan asortiman prodavnice prema kodovima jer ce
90
        pri kucanju racuna prodavac unositi kod artikla. Prilikom
        kucanja svakog racuna pretrazuje se asortiman, da bi se utvrdila
        cena artikla. Kucanje racuna obuhvata vise pretraga asortimana i
        cilj je da ta operacija bude sto efikasnija. Zato se koristi
94
        algoritam binarne pretrage prilikom pretrazivanja po kodu
        artikla. Iz tog razloga, potrebno je da asortiman bude sortiran
96
       po kodovima i to ce biti uradjeno primenom selection sort
       algoritma. Sortiranje se vrsi samo jednom na pocetku, ali se
98
        zato posle artikli mogu brzo pretrazivati prilikom kucanja
        proizvoljno puno racuna. Vreme koje se utrosi na sortiranje na
100
        pocetku izvrsavanja programa, kasnije se isplati jer se za
        brojna trazenja artikla umesto linearne moze koristiti
        efikasnija binarna pretraga. */
     selection_sort(asortiman, n);
104
     /* Ispis stanja u prodavnici */
106
     printf
         ("Asortiman:\nKOD
                                          Naziv artikla
                                                             Tme
108
       proizvodjaca
                          Cena\n");
    for (i = 0; i < n; i++)
       printf("%10ld %20s %20s %12.2f\n", asortiman[i].kod,
              asortiman[i].naziv, asortiman[i].proizvodjac,
              asortiman[i].cena);
    kod = 0;
114
    while (1) {
       printf("----\n");
       printf("- Za kraj za kraj rada kase, pritisnite CTRL+D!\n");
       printf("- Za nov racun unesite kod artikla:\n\n");
118
       /* Unos bar koda provog artikla sledeceg kupca */
       if (scanf("%ld", &kod) == EOF)
120
        break;
       /* Trenutni racun novog kupca */
       racun = 0;
       /* Za sve artikle trenutnog kupca */
124
       while (1) {
         /* Vrsi se njihov pronalazak u nizu */
         if ((i = binarna_pretraga(asortiman, n, kod)) == -1) {
           printf("\tGreska: Ne postoji proizvod sa trazenim kodom!\n");
128
         } else {
           printf("\tTrazili ste:\t%s %s %12.2f\tn",
130
                  asortiman[i].naziv, asortiman[i].proizvodjac,
                  asortiman[i].cena);
           /* I dodaje se cena na ukupan racun */
           racun += asortiman[i].cena;
134
```

```
136
         /* Unos bar koda sledeceg artikla trenutnog kupca, ili 0 ako
            on nema vise artikla */
         printf("Unesite kod artikla [ili 0 za prekid]: \t");
138
         scanf("%ld", &kod);
         if (kod == 0)
140
           break;
142
       /* Stampa se ukupan racun trenutnog kupca */
       printf("\n\tUKUPNO: %.21f dinara.\n\n", racun);
144
146
     printf("Kraj rada kase!\n");
148
     exit(EXIT_SUCCESS);
150 }
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
5 #define MAX 500
  /* Struktura sa svim informacijama o pojedinacnom studentu */
  typedef struct {
    char ime[21];
    char prezime [26];
    int prisustvo;
    int zadaci;
13 } Student;
15 /* Funkcija za sortiranje niza struktura po prezimenu leksikografski
     rastuce */
  void sort_ime_leksikografski(Student niz[], int n)
    int i, j;
    int min;
    Student pom;
21
    for (i = 0; i < n - 1; i++) {
      min = i;
      for (j = i + 1; j < n; j++)
        if (strcmp(niz[j].ime, niz[min].ime) < 0)</pre>
          min = j;
      if (min != i) {
29
        pom = niz[min];
        niz[min] = niz[i];
        niz[i] = pom;
```

```
}
  }
35
  /* Funkcija za sortiranje niza struktura po ukupnom broju uradjenih
     zadataka opadajuce, a ukoliko neki studenti imaju isti broj
     uradjenih zadataka sortiraju se po duzini imena rastuce. */
  void sort_zadatke_pa_imena(Student niz[], int n)
  {
41
    int i, j;
    int max;
43
    Student pom;
    for (i = 0; i < n - 1; i++) {
45
      max = i;
      for (j = i + 1; j < n; j++)
        if (niz[j].zadaci > niz[max].zadaci)
          max = j;
49
        else if (niz[j].zadaci == niz[max].zadaci
                 && strlen(niz[j].ime) < strlen(niz[max].ime))
          max = j;
      if (max != i) {
53
        pom = niz[max];
        niz[max] = niz[i];
        niz[i] = pom;
    }
  }
   /* Funkcija za sortiranje niza struktura po broju casova na kojima
      su bili opadajuce. Ukoliko neki studenti imaju isti broj casova,
      sortiraju se opadajuce po broju uradjenih zadataka, a ukoliko se
      i po broju zadataka poklapaju, njihovo sortiranje ce biti po
      prezimenu opadajuce. */
  void sort_prisustvo_pa_zadatke_pa_prezimena(Student niz[], int n)
    int i, j;
    int max;
    Student pom;
    for (i = 0; i < n - 1; i++) {
      max = i;
      for (j = i + 1; j < n; j++)
73
        if (niz[j].prisustvo > niz[max].prisustvo)
          max = j;
        else if (niz[j].prisustvo == niz[max].prisustvo
                 && niz[j].zadaci > niz[max].zadaci)
          max = j;
        else if (niz[j].prisustvo == niz[max].prisustvo
                 && niz[j].zadaci == niz[max].zadaci
                 && strcmp(niz[j].prezime, niz[max].prezime) > 0)
81
          max = j;
      if (max != i) {
83
        pom = niz[max];
85
        niz[max] = niz[i];
```

```
niz[i] = pom;
  }
89
  int main(int argc, char *argv[])
     Student praktikum[MAX];
93
     int i, br_studenata = 0;
95
     FILE *fp = NULL;
97
     /* Otvara se datoteka za citanje */
     if ((fp = fopen("aktivnost.txt", "r")) == NULL) {
99
       fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke aktivnost.txt.\n");
       exit(EXIT_FAILURE);
     /* Ucitava se sadrzaj */
     for (i = 0;
          fscanf(fp, "%s%s%d%d", praktikum[i].ime,
                 praktikum[i].prezime, &praktikum[i].prisustvo,
                 &praktikum[i].zadaci) != EOF; i++);
     /* Zatvara se datoteka */
     fclose(fp);
     br_studenata = i;
113
     /* Kreira se prvi spisak studenata po prvom kriterijumu */
     sort_ime_leksikografski(praktikum, br_studenata);
     /* Otvara se datoteka za pisanje */
     if ((fp = fopen("dat1.txt", "w")) == NULL) {
       fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke dat1.txt.\n");
119
       exit(EXIT_FAILURE);
    }
     /* Upisuje se niz u datoteku */
     fprintf
123
         (fp, "Studenti sortirani po imenu leksikografski rastuce:\n");
     for (i = 0; i < br_studenata; i++)</pre>
       praktikum[i].zadaci);
     /* Zatvara se datoteka */
     fclose(fp);
     /* Kreira se drugi spisak studenata po drugom kriterijumu */
     sort_zadatke_pa_imena(praktikum, br_studenata);
     /* Otvara se datoteka za pisanje */
     if ((fp = fopen("dat2.txt", "w")) == NULL) {
       fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke dat2.txt.\n");
```

```
exit(EXIT_FAILURE);
     }
139
     /* Upisuje se niz u datoteku */
     fprintf(fp, "Studenti sortirani po broju zadataka opadajuce,\n");
141
     fprintf(fp, "pa po duzini imena rastuce:\n");
     for (i = 0; i < br_studenata; i++)
       fprintf(fp, "%s %s %d %d\n", praktikum[i].ime,
               praktikum[i].prezime, praktikum[i].prisustvo,
               praktikum[i].zadaci);
     /* Zatvara se datoteka */
147
     fclose(fp);
149
     /* Kreira se treci spisak studenata po trecem kriterijumu */
     sort_prisustvo_pa_zadatke_pa_prezimena(praktikum, br_studenata);
     /* Otvara se datoteka za pisanje */
     if ((fp = fopen("dat3.txt", "w")) == NULL) {
       fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke dat3.txt.\n");
       exit(EXIT_FAILURE);
     }
     /* Upisuje se niz u datoteku */
     fprintf(fp, "Studenti sortirani po prisustvu opadajuce,\n");
159
     fprintf(fp, "pa po broju zadataka,\n");
     fprintf(fp, "pa po prezimenima leksikografski opadajuce:\n");
     for (i = 0; i < br_studenata; i++)</pre>
       fprintf(fp, "%s %s %d %d\n", praktikum[i].ime,
163
               praktikum[i].prezime, praktikum[i].prisustvo,
               praktikum[i].zadaci);
     /* Zatvara se datoteka */
     fclose(fp);
167
     exit(EXIT_SUCCESS);
```

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

#define KORAK 10

/* Struktura koja opisuje jednu pesmu */
typedef struct {
    char *izvodjac;
    char *naslov;
    int broj_gledanja;
} Pesma;

/* Funkcija za uporedjivanje pesama po broju gledanosti (potrebna za rad qsort funkcije) */
```

```
int uporedi_gledanost(const void *pp1, const void *pp2)
    Pesma *p1 = (Pesma *) pp1;
18
    Pesma *p2 = (Pesma *) pp2;
20
    return p2->broj_gledanja - p1->broj_gledanja;
22 }
  /* Funkcija za uporedjivanje pesama po naslovu (potrebna za rad
     qsort funkcije) */
26 int uporedi_naslove(const void *pp1, const void *pp2)
    Pesma *p1 = (Pesma *) pp1;
28
    Pesma *p2 = (Pesma *) pp2;
30
    return strcmp(p1->naslov, p2->naslov);
32 }
  /* Funkcija za uporedjivanje pesama po izvodjacu (potrebna za rad
     qsort funkcije) */
36 int uporedi_izvodjace(const void *pp1, const void *pp2)
    Pesma *p1 = (Pesma *) pp1;
38
    Pesma *p2 = (Pesma *) pp2;
40
    return strcmp(p1->izvodjac, p2->izvodjac);
42 }
  /* Funkcija koja oslobadja memoriju zauzetu dinamickim nizom pesme
     dimenzije n */
46
  void oslobodi(Pesma * pesme, int n)
48
    int i;
    for (i = 0; i < n; i++) {
     free(pesme[i].izvodjac);
     free(pesme[i].naslov);
    free(pesme);
54
  int main(int argc, char *argv[])
56
    FILE *ulaz;
                                   /* Pokazivac na deo memorije za
    Pesma *pesme;
                                      cuvanje pesama */
60
                                   /* Broj mesta alociranih za pesme */
    int alocirano_za_pesme;
                                   /* Redni broj pesme cije se
    int i;
62
                                      informacije citaju */
                                   /* Ukupan broj pesama */
    int n;
    int j;
66
    char c;
    int alocirano;
                                   /* Broj mesta alociranih za
```

```
68
                                       propratne informacije o pesmama */
     int broj_gledanja;
     /* Priprema se datoteka za citanje */
     ulaz = fopen("pesme.txt", "r");
72
     if (ulaz == NULL) {
       fprintf(stderr,
74
                "Greska: Neuspesno otvaranje ulazne datoteke!\n");
       exit(EXIT_FAILURE);
78
     /* Citaju se informacije o pesmama */
     pesme = NULL;
80
     alocirano_za_pesme = 0;
     i = 0;
82
     while (1) {
84
       /* Proverava se da li je dostignut kraj datoteke */
       c = fgetc(ulaz);
86
       if (c == EOF) {
         /* Nema vise sadrzaja za citanje */
88
         break;
       } else {
90
         /* Inace, vraca se procitani karakter nazad */
         ungetc(c, ulaz);
92
94
       /* Proverava se da li postoji dovoljno vec alocirane memorije za
          citanje nove pesme */
96
       if (alocirano_za_pesme == i) {
         /* Ako ne, ako je potrosena sva alocirana memorija, alocira se
            novih KORAK mesta */
         alocirano_za_pesme += KORAK;
100
         pesme =
             (Pesma *) realloc(pesme,
                                alocirano_za_pesme * sizeof(Pesma));
104
         /* Proverava se da li je nova memorija uspesno realocirana */
         if (pesme == NULL) {
106
           /* Ako nije ispisuje se obavestenje */
           fprintf(stderr, "Greska: Problem sa alokacijom memorije!\n");
108
           /* I oslobadja sva memorija zauzeta do ovog koraka */
           oslobodi(pesme, i);
           exit(EXIT_FAILURE);
         }
112
       }
114
       /* Ako jeste, nastavlja se sa citanjem pesama ... */
       /* Cita se ime izvodjaca */
116
       j = 0;
                                    /* Pozicija na koju treba smestiti
                                       procitani karakter */
118
       alocirano = 0;
                                    /* Broj alociranih mesta */
```

```
pesme[i].izvodjac = NULL;
                                    /* Memorija za smestanje procitanih
120
                                       karaktera */
       /* Sve do prve beline u liniji (beline koja se nalazi nakon
          imena izvodjaca) citaju se karakteri iz datoteke */
124
       while ((c = fgetc(ulaz)) != ' ') {
         /* Provera da li postoji dovoljno memorije za smestanje
126
            procitanog karaktera */
         if (j == alocirano) {
128
           /* Ako ne, ako je potrosena sva alocirana memorija, alocira
130
              se novih KORAK mesta */
           alocirano += KORAK:
           pesme[i].izvodjac =
               (char *) realloc(pesme[i].izvodjac,
                                 alocirano * sizeof(char));
136
           /* Provera da li je nova alokacija uspesna */
           if (pesme[i].izvodjac == NULL) {
138
             /* Ako nije oslobadja se sva memorija zauzeta do ovog
                koraka */
140
             oslobodi(pesme, i);
             /* I prekida sa izvrsavanjem programa */
             exit(EXIT_FAILURE);
           }
144
         }
         /* Ako postoji dovoljno alocirane memorije, smesta se vec
146
            procitani karakter */
         pesme[i].izvodjac[j] = c;
148
         j++;
         /* I nastavlja se sa citanjem */
       /* Upis terminirajuce nule na kraj reci */
154
       pesme[i].izvodjac[j] = '\0';
       /* Preskace se karakter - */
       fgetc(ulaz);
158
       /* Preskace se razmak */
       fgetc(ulaz);
       /* Cita se naslov pesme */
       i = 0;
                                    /* Pozicija na koju treba smestiti
                                       procitani karakter */
164
       alocirano = 0;
                                    /* Broj alociranih mesta */
       pesme[i].naslov = NULL;
                                    /* Memorija za smestanje procitanih
                                       karaktera */
168
       /* Sve do zareza (koji se nalazi nakon naslova pesme) citaju se
          karakteri iz datoteke */
       while ((c = fgetc(ulaz)) != ',') {
```

```
/* Provera da li postoji dovoljno memorije za smestanje
            procitanog karaktera */
         if (j == alocirano) {
174
           /* Ako ne, ako je potrosena sva alocirana memorija, alocira
              se novih KORAK mesta */
           alocirano += KORAK:
           pesme[i].naslov =
178
               (char *) realloc(pesme[i].naslov,
                                 alocirano * sizeof(char));
180
           /* Provera da li je nova alokacija uspesna */
182
           if (pesme[i].naslov == NULL) {
             /* Ako nije, oslobadja se sva memorija zauzeta do ovog
184
                koraka */
             free(pesme[i].izvodjac);
186
             oslobodi(pesme, i);
             /* I prekida izvrsavanje programa */
188
             exit(EXIT_FAILURE);
           }
190
         }
         /* Smesta se procitani karakter */
         pesme[i].naslov[j] = c;
         j++;
194
         /* I nastavlja dalje sa citanjem */
196
       /* Upisuje se terminirajuca nula na kraj reci */
       pesme[i].naslov[j] = '\0';
198
       /* Preskace se razmak */
200
       fgetc(ulaz);
       /* Cita se broj gledanja */
       broj_gledanja = 0;
204
       /* Sve do znaka za novi red (kraja linije) citaju se karakteri
206
          iz datoteke */
       while ((c = fgetc(ulaz)) != '\n') {
208
         broj_gledanja = broj_gledanja * 10 + (c - '0');
210
       pesme[i].broj_gledanja = broj_gledanja;
212
       /* Prelazi se na citanje sledece pesme */
214
       i++;
     }
     /* Informacija o broju procitanih pesama */
     n = i:
218
     /* Zatvara se datoteka */
     fclose(ulaz);
     /* Analiza argumenta komandne linije */
     if (argc == 1) {
```

```
/* Nema dodatnih opcija => sortiranje po broju gledanja */
224
       qsort(pesme, n, sizeof(Pesma), &uporedi_gledanost);
     } else {
226
       if (argc == 2 && strcmp(argv[1], "-n") == 0) {
         /* Sortira se po naslovu */
228
         qsort(pesme, n, sizeof(Pesma), &uporedi_naslove);
       } else {
230
         if (argc == 2 && strcmp(argv[1], "-i") == 0) {
           /* Sortira se po izvodjacu */
           qsort(pesme, n, sizeof(Pesma), &uporedi_izvodjace);
         } else {
234
           fprintf(stderr, "Greska: Nedozvoljeni argumenti!\n");
           free(pesme);
236
           exit(EXIT_FAILURE);
238
       }
240
     /* Ispis rezultata */
242
     for (i = 0; i < n; i++) {
       printf("%s - %s, %d\n", pesme[i].izvodjac, pesme[i].naslov,
244
              pesme[i].broj_gledanja);
     /* Oslobadja se memorija */
248
     oslobodi(pesme, n);
250
     exit(EXIT_SUCCESS);
  }
252
```

NAPOMENA: Rešenje koristi biblioteku za rad sa celobrojnim matricama iz zadatka 2.19.

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

/* Funkcija koja odredjuje zbir v-te vrste matrice a koja ima m
kolona */
int zbir_vrste(int **a, int v, int m)
{
   int i, zbir = 0;

for (i = 0; i < m; i++) {
   zbir += a[v][i];
   }
  return zbir;
}</pre>
```

```
/* Funkcija koja sortira vrste matrice (pokazivace na vrste) na
    osnovu zbira koriscenjem selection sort algoritma */
  void sortiraj_vrste(int **a, int n, int m)
20 {
    int i, j, min;
    for (i = 0; i < n - 1; i++) {
      min = i:
24
      for (j = i + 1; j < n; j++) {
        if (zbir_vrste(a, j, m) < zbir_vrste(a, min, m)) {</pre>
26
          min = j;
28
      if (min != i) {
30
        int *tmp;
        tmp = a[i];
        a[i] = a[min];
        a[min] = tmp;
34
36
38
  int main(int argc, char *argv[])
40 {
    int **a;
    int n, m;
42
    /* Unos dimenzija matrice */
44
    printf("Unesite dimenzije matrice: ");
    scanf("%d %d", &n, &m);
46
    /* Alokacija memorije */
48
    a = alociraj_matricu(n, m);
    if (a == NULL) {
      fprintf(stderr, "Greska: Neuspesna alokacija matrice\n");
      exit(EXIT_FAILURE);
54
    /* Ucitavaju se elementi matrice */
    printf("Unesite elemente matrice po vrstama:\n");
56
    ucitaj_matricu(a, n, m);
58
    /* Poziva se funkcija koja sortira vrste matrice prema zbiru */
    sortiraj_vrste(a, n, m);
60
    /* Ispisuje se rezultujuca matrica */
    printf("Sortirana matrica je:\n");
    ispisi_matricu(a, n, m);
    /* Oslobadja se memorija */
    a = dealociraj_matricu(a, n);
68
```

```
exit(EXIT_SUCCESS);
70 }
```

```
1 #include <stdio.h>
  #include <stdlib.h>
  #include <math.h>
  #include <search.h>
  #define MAX 100
  /* Funkcija poredjenja dva cela broja (neopadajuci poredak) */
9 int poredi_int(const void *a, const void *b)
    /* Potrebno je konvertovati void pokazivace u int pokazivace koji
       se zatim dereferenciraju. Vraca se razlika dobijenih int-ova. */
    int b1 = *((int *) a);
    int b2 = *((int *) b);
    /* Zbog moguceg prekoracenja opsega celih brojeva, oduzimanje
       b1-b2 treba izbegavati */
    if (b1 > b2)
      return 1;
    else if (b1 < b2)
21
      return -1;
    else
      return 0;
23
  /* Funkcija poredjenja dva cela broja (nerastuci poredak) */
27 int poredi_int_nerastuce(const void *a, const void *b)
    /* Za obrnuti poredak treba samo promeniti znak vrednosti koju
       koju vraca prethodna funkcija */
    return -poredi_int(a, b);
33
  int main()
    size_t n;
    int i, x;
    int a[MAX], *p = NULL;
39
    /* Unos dimenzije */
    printf("Uneti dimenziju niza: ");
41
    scanf("%ld", &n);
    if (n > MAX)
43
      n = MAX;
45
    /* Unos elementa niza */
```

```
printf("Uneti elemente niza:\n");
    for (i = 0; i < n; i++)
      scanf("%d", &a[i]);
49
    /* Sortira se niz celih brojeva */
    qsort(a, n, sizeof(int), &poredi_int);
53
    /* Prikazuje se sortirani niz */
    printf("Sortirani niz u rastucem poretku:\n");
    for (i = 0; i < n; i++)
     printf("%d ", a[i]);
    putchar('\n');
    /* Pretrazuje se niz */
    /* Vrednost koja ce biti trazena u nizu */
    printf("Uneti element koji se trazi u nizu: ");
    scanf("%d", &x);
    /* Binarna pretraga */
    printf("Binarna pretraga: \n");
    p = bsearch(&x, a, n, sizeof(int), &poredi_int);
    if (p == NULL)
     printf("Elementa nema u nizu!\n");
    else
     printf("Element je nadjen na poziciji %ld\n", p - a);
71
    /* Linearna pretraga */
73
    printf("Linearna pretraga (lfind): \n");
    p = lfind(&x, a, &n, sizeof(int), &poredi_int);
    if (p == NULL)
      printf("Elementa nema u nizu!\n");
    else
      printf("Element je nadjen na poziciji %ld\n", p - a);
79
81
    return 0;
```

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

#define MAX 100

/* Funkcija racuna broj delilaca broja x */
int broj_delilaca(int x)
{
   int i;
   int br;
}
```

```
13
    /* Negativni brojevi imaju isti broj delilaca kao i pozitivni */
    if (x < 0)
      x = -x;
    if (x == 0)
17
      return 0;
    if (x == 1)
19
      return 1;
    /* Svaki broj veci od 1 ima bar 2 delioca, (1 i samog sebe) */
    br = 2;
    for (i = 2; i < sqrt(x); i++)
      if (x % i == 0)
        /* Ako i deli x onda su delioci: i, x/i */
        br += 2;
    /* Ako je broj x bas kvadrat, onda se iz petlje izaslo kada je
       promenljiva i bila bas jednaka korenu od x, i tada broj x ima
       jos jednog delioca */
29
    if (i * i == x)
      br++;
31
    return br;
33
35
  /* Funkcija poredjenja dva cela broja po broju delilaca */
  int poredi_po_broju_delilaca(const void *a, const void *b)
37
    int ak = *(int *) a;
39
    int bk = *(int *) b;
    int n_d_a = broj_delilaca(ak);
41
    int n_d_b = broj_delilaca(bk);
43
    return n_d_a - n_d_b;
  }
45
47
  int main()
49
    size_t n;
    int i;
    int a[MAX];
    /* Unos dimenzije */
    printf("Uneti dimenziju niza: ");
    scanf("%ld", &n);
    if (n > MAX)
      n = MAX;
    /* Unos elementa niza */
59
    printf("Uneti elemente niza:\n");
    for (i = 0; i < n; i++)
61
      scanf("%d", &a[i]);
63
    /* Sortira se niz celih brojeva prema broju delilaca */
```

```
qsort(a, n, sizeof(int), &poredi_po_broju_delilaca);

/* Prikazuje se sortirani niz */
printf("Sortirani niz u rastucem poretku prema broju delilaca:\n");
for (i = 0; i < n; i++)
    printf("%d ", a[i]);
putchar('\n');

return 0;
}</pre>
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
  #include <search.h>
  #define MAX_NISKI 1000
7 #define MAX_DUZINA 31
 Niz nizova karaktera ovog potpisa
  char niske[3][4];
  se moze graficki predstaviti ovako:
   | a | b | c | \0 | | d | e | \0 | | | f | g | h | \0 | |
   Dakle kao tri reci (abc, de, fgh), nadovezane jedna na drugu. Za
  svaku je rezervisano po 4 karaktera ukljucujuci \0. Druga rec sa
  nalazi na adresi koja je za 4 veca od prve reci, a za 4 manja od
  adrese na kojoj se nalazi treca rec. Adresa i-te reci je niske[i]
   i ona je tipa char*.
21
   Kako pokazivaci a i b u sledecoj funkciji sadrze adrese elemenata
23 koji trebaju biti uporedjeni, (npr. pri porecenju prve i poslednje
  reci, pokazivac a ce pokazivati na slovo 'a', a pokazivac b na
  slovo 'f') treba ih kastovati na char*, i pozvati funkciju strcmp
  nad njima.
int poredi_leksikografski(const void *a, const void *b)
   return strcmp((char *) a, (char *) b);
31 }
33 /* Funkcija slicna prethodnoj, osim sto elemente ne uporedjuje
    leksikografski, vec po duzini */
int poredi_duzine(const void *a, const void *b)
   return strlen((char *) a) - strlen((char *) b);
```

```
int main()
  {
41
    int i;
    size_t n;
43
    FILE *fp = NULL;
    char niske[MAX_NISKI][MAX_DUZINA];
45
    char *p = NULL;
    char x[MAX_DUZINA];
47
    /* Otvara se datoteka */
49
    if ((fp = fopen("niske.txt", "r")) == NULL) {
      fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke niske.txt.\n");
      exit(EXIT_FAILURE);
53
    /* Cita se sadrzaj datoteke */
    for (i = 0; fscanf(fp, "%s", niske[i]) != EOF; i++);
    /* Zatvara se datoteka */
    fclose(fp);
    n = i;
61
    /* Sortiraju se niske leksikografski. Biblioteckoj funkciji qsort
       prosledjuje se funkcija kojom se zadaje kriterijum poredjenja 2
       niske po duzini */
    qsort(niske, n, MAX_DUZINA * sizeof(char), &poredi_leksikografski);
67
    printf("Leksikografski sortirane niske:\n");
    for (i = 0; i < n; i++)
      printf("%s ", niske[i]);
    printf("\n");
73
    /* Unosi se trazena niska */
    printf("Uneti trazenu nisku: ");
    scanf("%s", x);
    /* Binarna pretraga */
    /* Prosledjuje se pokazivac na funkciju poredi_leksikografski jer
       je niz vec sortiran leksikografski. */
    p = bsearch(&x, niske, n, MAX_DUZINA * sizeof(char),
                 &poredi_leksikografski);
81
    if (p != NULL)
83
      printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
             p, (p - (char *) niske) / MAX_DUZINA);
85
    else
      printf("Niska nije pronadjena u nizu\n");
87
    /* Sortira se po duzini */
89
    qsort(niske, n, MAX_DUZINA * sizeof(char), &poredi_duzine);
```

```
91
     printf("Niske sortirane po duzini:\n");
    for (i = 0; i < n; i++)
93
     printf("%s ", niske[i]);
     printf("\n");
95
     /* Linearna pretraga */
97
     p = lfind(&x, niske, &n, MAX_DUZINA * sizeof(char),
               &poredi_leksikografski);
99
     if (p != NULL)
      printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
              p, (p - (char *) niske) / MAX_DUZINA);
       printf("Niska nije pronadjena u nizu\n");
     exit(EXIT_SUCCESS);
107 }
```

```
1 #include <stdio.h>
 #include <stdlib.h>
 #include <string.h>
 #include <search.h>
  #define MAX_NISKI 1000
7 #define MAX_DUZINA 31
 Niz pokazivaca na karaktere ovog potpisa
   char *niske[3];
   posle alokacije u main-u se moze graficki predstaviti ovako:
13
    | X | -----> | a | b | c | \0|
                          ===========
    | Y | ----->
                        | d | e | \0|
                          _____
17
    | Z | ------ | f | g | h | \0|
19
   Sa leve strane je vertikalno prikazan niz pokazivaca, gde je i-ti
21
   njegov element pokazivac koji pokazuje na alocirane karaktere i-te
   reci. Njegov tip je char*.
   Kako pokazivaci a i b u sledecoj funkciji sadrze adrese elemenata
   koji trebaju biti uporedjeni (recimo adresu od X i adresu od Z), i
25
   kako su X i Z tipa char*, onda a i b su tipa char**, pa se tako
   moraju i kastovati. Da bi se leksikografski uporedili elementi X i
27
   Z, moraju se uporediti stringovi na koje oni pokazuju, pa zato se
   u sledecoj funkciji poziva strcmp() nad onim na sta pokazuju a i b,
29
   kastovani na odgovarajuci tip.
```

```
int poredi_leksikografski(const void *a, const void *b)
33
    return strcmp(*(char **) a, *(char **) b);
35 }
  /* Funkcija slicna prethodnoj, osim sto elemente ne uporedjuje
     leksikografski, vec po duzini */
int poredi_duzine(const void *a, const void *b)
    return strlen(*(char **) a) - strlen(*(char **) b);
41
43
  /* Ovo je funkcija poredjenja za bsearch. Pokazivac b pokazuje na
     element u nizu sa kojim se poredi, pa njega treba kastovati na
45
     char** i dereferencirati, (videti obrazlozenje za prvu funkciju u
     ovom zadatku, a pokazivac a pokazuje na element koji se trazi. U
47
     main funkciji je to x, koji je tipa char*, tako da pokazivac a
     ovde samo treba kastovati i ne dereferencirati. */
49
  int poredi_leksikografski_b(const void *a, const void *b)
51 {
    return strcmp((char *) a, *(char **) b);
53 }
55 int main()
    int i;
    size_t n;
    FILE *fp = NULL;
59
    char *niske[MAX_NISKI];
    char **p = NULL;
61
    char x[MAX_DUZINA];
63
    /* Otvara se datoteka */
    if ((fp = fopen("niske.txt", "r")) == NULL) {
      fprintf(stderr,
               "Greska: Neupesno otvaranje datoteke niske.txt.\n");
      exit(EXIT_FAILURE);
    /* Cita se sadrzaj datoteke */
    i = 0;
    while (fscanf(fp, "%s", x) != EOF) {
73
      /* Alocira se dovoljno memorije za i-tu nisku */
      if ((niske[i] = malloc((strlen(x) + 1) * sizeof(char))) == NULL)
        fprintf(stderr, "Greska: Neuspesna alokacija niske\n");
        exit(EXIT_FAILURE);
      /* Kopira se procitana niska na svoje mesto */
      strcpy(niske[i], x);
      i++;
```

```
83
     /* Zatvara se datoteka */
     fclose(fp);
85
     n = i;
87
     /* Sortiraju se niske leksikografski. Biblioteckoj funkciji qsort
        se prosledjuje funkcija kojom se zadaje kriterijum poredjenja 2
89
        niske po duzini */
    qsort(niske, n, sizeof(char *), &poredi_leksikografski);
91
     printf("Leksikografski sortirane niske:\n");
93
    for (i = 0; i < n; i++)
      printf("%s ", niske[i]);
95
     printf("\n");
97
     /* Unosi se trazena niska */
     printf("Uneti trazenu nisku: ");
99
     scanf("%s", x);
     /* Binarna pretraga */
     p = bsearch(x, niske, n, sizeof(char *), &poredi_leksikografski_b);
     if (p != NULL)
       printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
              *p, p - niske);
       printf("Niska nije pronadjena u nizu\n");
     /* Linearna pretraga */
     p = lfind(x, niske, &n, sizeof(char *), &poredi_leksikografski_b);
     if (p != NULL)
       printf("Niska \"%s\" je pronadjena u nizu na poziciji %ld\n",
113
              *p, p - niske);
    else
       printf("Niska nije pronadjena u nizu\n");
     /* Sortira se po duzini */
     qsort(niske, n, sizeof(char *), &poredi_duzine);
     printf("Niske sortirane po duzini:\n");
    for (i = 0; i < n; i++)
121
      printf("%s ", niske[i]);
    printf("\n");
    /* Oslobadja se zauzeta memorija */
    for (i = 0; i < n; i++)
      free(niske[i]);
127
    exit(EXIT_SUCCESS);
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include <search.h>
  #define MAX 500
  /* Struktura sa svim informacijama o pojedinacnom studentu */
  typedef struct {
    char ime[21];
    char prezime[21];
    int bodovi;
  } Student;
  /* Funkcija poredjenja za sortiranje po broju bodova. Studenti sa
    istim brojem bodova se dodatno sortiraju leksikografski po
     prezimenu */
int poredi1(const void *a, const void *b)
    Student *prvi = (Student *) a;
    Student *drugi = (Student *) b;
    if (prvi->bodovi > drugi->bodovi)
      return -1;
    else if (prvi->bodovi < drugi->bodovi)
    else
      /* Ako su jednaki po broju bodova, treba ih uporediti po
         prezimenu */
      return strcmp(prvi->prezime, drugi->prezime);
  /* Funkcija za poredjenje koja se koristi u pretrazi po broju
    bodova. Prvi parametar je ono sto se trazi u nizu (broj bodova),
     a drugi parametar je element niza ciji se bodovi porede. */
int poredi2(const void *a, const void *b)
    int bodovi = *(int *) a;
    Student *s = (Student *) b;
    return s->bodovi - bodovi;
  }
42
  /* Funkcija za poredjenje koja se koristi u pretrazi po prezimenu.
     Prvi parametar je ono sto se trazi u nizu (prezime), a drugi
     parametar je element niza cije se prezime poredi. */
46 int poredi3(const void *a, const void *b)
    char *prezime = (char *) a;
48
    Student *s = (Student *) b;
    return strcmp(prezime, s->prezime);
```

```
| }
  int main(int argc, char *argv[])
54 {
    Student kolokvijum [MAX];
    int i:
56
    size_t br_studenata = 0;
    Student *nadjen = NULL;
58
    FILE *fp = NULL;
    int bodovi;
    char prezime[21];
    /* Ako je program pozvan sa nedovoljnim brojem argumenata daje se
       informacija korisniku kako se program koristi i prekida se
64
       izvrsavanje. */
    if (argc < 2) {
      fprintf(stderr, "Greska: Program se poziva sa %s datoteka\n",
               argv[0]);
68
      exit(EXIT_FAILURE);
    }
    /* Otvara se datoteka */
    if ((fp = fopen(argv[1], "r")) == NULL) {
      fprintf(stderr, "Greska: Neupesno otvaranje datoteke %s\n",
74
               argv[1]);
      exit(EXIT_FAILURE);
78
    /* Ucitava se sadrzaj */
    for (i = 0;
80
         fscanf(fp, "%s%s%d", kolokvijum[i].ime,
                 kolokvijum[i].prezime,
82
                 &kolokvijum[i].bodovi) != EOF; i++);
84
    /* Zatvara se datoteka */
    fclose(fp);
86
    br_studenata = i;
88
    /* Sortira se niz studenata po broju bodova, gde se unutar grupe
       studenata sa istim brojem bodova sortiranje vrsi po prezimenu */
90
    qsort(kolokvijum, br_studenata, sizeof(Student), &poredi1);
    printf("Studenti sortirani po broju poena opadajuce, ");
    printf("pa po prezimenu rastuce:\n");
94
    for (i = 0; i < br_studenata; i++)</pre>
      printf("%s %s %d\n", kolokvijum[i].ime,
96
              kolokvijum[i].prezime, kolokvijum[i].bodovi);
98
    /* Pretrazuju se studenati po broju bodova binarnom pretragom jer
       je niz sortiran po broju bodova. */
    printf("Unesite broj bodova: ");
    scanf("%d", &bodovi);
```

```
nadjen =
104
         bsearch(&bodovi, kolokvijum, br_studenata, sizeof(Student),
                  &poredi2);
106
     if (nadjen != NULL)
108
       printf
           ("Pronadjen je student sa unetim brojem bodova: %s %s %d\n",
            nadjen->ime, nadjen->prezime, nadjen->bodovi);
     else
       printf("Nema studenta sa unetim brojem bodova\n");
114
     /* Pretraga po prezimenu se mora vrsiti linearno jer je niz
        sortiran po bodovima. */
     printf("Unesite prezime: ");
     scanf("%s", prezime);
118
     nadjen =
         lfind(prezime, kolokvijum, &br_studenata, sizeof(Student),
               &poredi3);
     if (nadjen != NULL)
124
       printf
           ("Pronadjen je student sa unetim prezimenom: %s %s %d\n",
126
            nadjen->ime, nadjen->prezime, nadjen->bodovi);
128
       printf("Nema studenta sa unetim prezimenom\n");
130
     exit(EXIT_SUCCESS);
132 }
```

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

#define MAX 128

/* Funkcija poredi dva karaktera */
int uporedi_char(const void *pa, const void *pb)
{
    return *(char *) pa - *(char *) pb;
}

/* Funkcija vraca 1 ako su argumenti anagrami, a 0 inace */
int anagrami(char s[], char t[])
{
    /* Ako dve niske imaju razlicitu duzinu onda one nisu anagrami */
    if (strlen(s) != strlen(t))
    return 0;
```

```
/* Sortiraju se karakteri u niskama */
20
    qsort(s, strlen(s) / sizeof(char), sizeof(char), &uporedi_char);
    qsort(t, strlen(t) / sizeof(char), sizeof(char), &uporedi_char);
    /* Ako su niske nakon sortiranja iste onda one jesu anagrami, u
       suprotnom, nisu */
    return !strcmp(s, t);
26
28
  int main()
30 {
    char s[MAX], t[MAX];
    /* Unose se niske */
    printf("Unesite prvu nisku: ");
34
    scanf("%s", s);
    printf("Unesite drugu nisku: ");
36
    scanf("%s", t);
38
    /* Ispituje se da li su niske anagrami */
    if (anagrami(s, t))
40
     printf("jesu\n");
    else
42
      printf("nisu\n");
44
    return 0;
 }
46
```

```
| #include <stdio.h>
  #include <string.h>
3 #include <stdlib.h>
5 #define MAX 10
  #define MAX_DUZINA 32
  /* Funkcija poredjenja */
9 int uporedi_niske(const void *pa, const void *pb)
   return strcmp((char *) pa, (char *) pb);
  }
13
  int main()
15 {
    int i, n;
   char S[MAX][MAX_DUZINA];
17
   /* Unosi se broj niski */
19
    printf("Unesite broj niski:");
```

```
scanf("%d", &n);
    /* Unosi se niz niski */
23
    printf("Unesite niske:\n");
    for (i = 0; i < n; i++)
25
     scanf("%s", S[i]);
    /* Sortira se niz niski */
    qsort(S, n, MAX_DUZINA * sizeof(char), &uporedi_niske);
    /************************
31
     Ovaj deo je iskomentarisan jer se u zadatku ne trazi ispis
     sortiranih niski. Koriscen je samo u fazi testiranja programa.
     printf("Sortirane niske su:\n");
35
     for(i = 0; i < n; i++)
       printf("%s ", S[i]);
37
    **********************
    /* Ako postoje dve iste niske u nizu, onda ce one nakon sortiranja
      niza biti jedna do druge */
41
    for (i = 0; i < n - 1; i++)
     if (strcmp(S[i], S[i + 1]) == 0) {
43
       printf("ima\n");
       return 0;
45
    printf("nema\n");
   return 0;
49
```

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

#define MAX 21

/* Struktura koja predstavlja jednog studenta */
typedef struct student {
    char nalog[8];
    char ime[MAX];
    char prezime[MAX];
    int poeni;
} Student;

/* Funkcija poredi studente prema broju poena, rastuce */
int uporedi_poeni(const void *a, const void *b)

{
    Student s = *(Student *) a;
```

```
Student t = *(Student *) b;
    return s.poeni - t.poeni;
21 }
23 /* Funkcija poredi studente prvo prema godini, zatim prema smeru i
    na kraju prema indeksu */
int uporedi_nalog(const void *a, const void *b)
    Student s = *(Student *) a;
27
    Student t = *(Student *) b;
    /* Za svakog studenta iz naloga se izdvaja godina upisa, smer i
      broj indeksa */
    int godina1 = (s.nalog[2] - '0') * 10 + s.nalog[3] - '0';
    int godina2 = (t.nalog[2] - '0') * 10 + t.nalog[3] - '0';
   char smer1 = s.nalog[1];
33
   char smer2 = t.nalog[1];
   int indeks1 =
35
        (s.nalog[4] - '0') * 100 + (s.nalog[5] - '0') * 10 +
       s.nalog[6] - '0';
   int indeks2 =
        (t.nalog[4] - '0') * 100 + (t.nalog[5] - '0') * 10 +
        t.nalog[6] - '0';
    if (godina1 != godina2)
41
     return godina1 - godina2;
    else if (smer1 != smer2)
43
     return smer1 - smer2;
    else
45
      return indeks1 - indeks2;
47 }
49 /* Funkcija poredjenja po nalogu za upotrebu u biblioteckoj funkciji
     bsearch */
int uporedi_bsearch(const void *a, const void *b)
53
   /* Nalog studenta koji se trazi */
   char *nalog = (char *) a;
    /* Kljuc pretrage */
    Student s = *(Student *) b;
    int godina1 = (nalog[2] - '0') * 10 + nalog[3] - '0';
   int godina2 = (s.nalog[2] - '0') * 10 + s.nalog[3] - '0';
59
    char smer1 = nalog[1];
    char smer2 = s.nalog[1];
    int indeks1 =
        (nalog[4] - '0') * 100 + (nalog[5] - '0') * 10 + (nalog[6] -
    int indeks2 =
        (s.nalog[4] - '0') * 100 + (s.nalog[5] - '0') * 10 +
        (s.nalog[6] - '0');
    if (godina1 != godina2)
     return godina1 - godina2;
    else if (smer1 != smer2)
```

```
return smer1 - smer2;
     else
       return indeks1 - indeks2;
   int main(int argc, char **argv)
  {
77
     Student *nadjen = NULL;
     char nalog_trazeni[8];
     Student niz_studenata[100];
     int i = 0, br_studenata = 0;
81
     FILE *in = NULL, *out = NULL;
83
     /* Ako je broj argumenata komandne linije razlicit i od 2 i od 3,
        korisnik nije ispravno pokrenuo program. */
85
     if (argc != 2 && argc != 3) {
       fprintf(stderr,
87
               "Greska: Program se poziva sa %s -opcija [nalog]\n",
               argv[0]);
89
       exit(EXIT_FAILURE);
91
     /* Otvara se datoteka za citanje */
93
     in = fopen("studenti.txt", "r");
     if (in == NULL) {
95
       fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke studenti.txt!\n");
97
       exit(EXIT_FAILURE);
99
     /* Otvara se datoteka za pisanje */
     out = fopen("izlaz.txt", "w");
     if (out == NULL) {
       fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke izlaz.txt!\n");
       exit(EXIT_FAILURE);
     /* Ucitavaju se studenti iz ulazne datoteke sve do njenog kraja */
     while (fscanf
            (in, "%s %s %s %d", niz_studenata[i].nalog,
             niz_studenata[i].ime, niz_studenata[i].prezime,
             &niz_studenata[i].poeni) != EOF)
113
       i++;
     br_studenata = i;
117
     /* Ako je prisutna opcija -p, vrsi se sortiranje po poenima */
     if (strcmp(argv[1], "-p") == 0)
119
       qsort(niz_studenata, br_studenata, sizeof(Student),
             &uporedi_poeni);
     /* A ako je prisutna opcija -n, vrsi se sortiranje po nalogu */
```

```
else if (strcmp(argv[1], "-n") == 0)
       qsort(niz_studenata, br_studenata, sizeof(Student),
             &uporedi_nalog);
     /* Sortirani studenti se ispisuju u izlaznu datoteku */
127
     for (i = 0; i < br_studenata; i++)</pre>
       fprintf(out, "%s %s %s %d\n", niz_studenata[i].nalog,
               niz_studenata[i].ime, niz_studenata[i].prezime,
               niz_studenata[i].poeni);
     /* Ukoliko je u komandnoj liniji uz opciju -n naveden i nalog
        studenta... */
     if (argc == 3 \&\& (strcmp(argv[1], "-n") == 0)) {
       strcpy(nalog_trazeni, argv[2]);
       /* ... pronalazi se student sa tim nalogom. */
       nadjen =
           (Student *) bsearch(nalog_trazeni, niz_studenata,
                                br_studenata, sizeof(Student),
141
                                &uporedi_bsearch);
       if (nadjen == NULL)
         printf("Nije nadjen!\n");
145
       else
         printf("%s %s %s %d\n", nadjen->nalog, nadjen->ime,
147
                nadjen->prezime, nadjen->poeni);
149
     /* Zatvaraju se datoteke */
     fclose(in);
     fclose(out);
     exit(EXIT_SUCCESS);
```

# 4

# Dinamičke strukture podataka

### 4.1 Liste

Zadatak 4.1 Napisati biblioteku za rad sa jednostruko povezanom listom čiji čvorovi sadrže cele brojeve.

- (a) Definisati strukturu Cvor kojom se predstavlja čvor liste. Čvor treba da sadrži ceo broj vrednost i pokazivač na sledeći čvor liste.
- (b) Napisati funkciju Cvor \*napravi\_cvor(int broj) koja kao argument dobija ceo broj, kreira nov čvor liste, inicijalizuje mu polja i vraća njegovu adresu.
- (c) Napisati funkciju int dodaj\_na\_pocetak\_liste(Cvor \*\* adresa\_glave, int broj) koja dodaje novi čvor sa vrednošću broj na početak liste, čija glava se nalazi na adresi adresa\_glave.
- (d) Napisati funkciju Cvor \*pronadji\_poslednji(Cvor \* glava) koja pronalazi poslednji čvor u listi.
- (e) Napisati funkciju int dodaj\_na\_kraj\_liste(Cvor \*\* adresa\_glave, int broj) koja dodaje novi čvor sa vrednošću broj na kraj liste.
- (f) Napisati funkciju Cvor \*pronadji\_mesto\_umetanja(Cvor \* glava, int broj) koja vraća pokazivač na čvor u neopadajuće uređenoj listi iza kojeg bi trebalo dodati nov čvor sa vrednošću broj.

- (g) Napisati funkciju int dodaj\_iza(Cvor \* tekuci, int broj) koja iza čvora tekuci dodaje novi čvor sa vrednošću broj.
- (h) Napisati funkciju int dodaj\_sortirano(Cvor \*\* adresa\_glave, int broj) koja dodaje novi elemenat u neopadajuće uređenu listu tako da se očuva postojeće uređenje.
- (i) Napisati funkciju void ispisi\_listu(Cvor \* glava) koja ispisuje čvorove liste uokvirene zagradama [, ] i međusobno razdvojene zapetama.
- (j) Napisati funkciju Cvor \*pretrazi\_listu(Cvor \* glava, int broj) koja proverava da li se u listi nalazi čvor čija je vrednost jednaka argumentu broj. Funkcija vraća pokazivač na pronađeni čvor ili NULL ukoliko ga ne pronađe.
- (k) Napisati funkciju Cvor \*pretrazi\_sortiranu\_listu(Cvor \* glava, int broj) koja proverava da li se u listi nalazi čvor sa vrednošću broj, pri čemu se pretpostavlja da se pretražuje neopadajuće uređena lista.
- (l) Napisati funkciju void obrisi\_cvor(Cvor \*\* adresa\_glave, int broj) koja briše sve čvorove u listi koji imaju vrednost jednaku argumentu broj.
- (m) Napisati funkciju void obrisi\_cvor\_sortirane\_liste(Cvor \*\* adresa-\_glave, int broj) koja briše sve čvorove u listi koji imaju vrednost jednaku argumentu broj, pri čemu se pretpostavlja da se briše iz neopadajuće uređene liste.
- (n) Napisati funkciju void oslobodi\_listu(Cvor \*\* adresa\_glave) koja oslobađa dinamički zauzetu memoriju za čvorove liste.

Funkcije dodavanja novog elementa u postojeću listu poput, dodaj\_na\_poce-tak\_liste, dodaj\_na\_kraj\_liste i dodaj\_sortirano, treba da vrate 0, ukoliko je sve bilo u redu, odnosno 1, ukoliko se dogodila greška prilikom alokacije memorije za nov čvor. NAPOMENA: Sve funkcije za rad sa listom implementirati iterativno.

Napisati programe koji koriste jednostruko povezanu listu za čuvanje elemenata koji se unose sa standardnog ulaza. Unošenje novih brojeva u listu prekida se učitavanjem kraja ulaza (EOF). Svako dodavanje novog broja u listu ispratiti ispisivanjem trenutnog sadržaja liste.

(1) U programu se učitani celi brojevi dodaju na početak liste. Unosi se ceo broj koji se traži u unetoj listi i na ekran se ispisuje rezultat pretrage.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
2
Lista: [2]
3
Lista: [3, 2]
14
Lista: [14, 3, 2]
5
Lista: [5, 14, 3, 2]
3
Lista: [3, 5, 14, 3, 2]
17
Lista: [17, 3, 5, 14, 3, 2]
Unesite broj koji se trazi: 5
Trazeni broj 5 je u listi!
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
23
Lista: [23]
14
Lista: [14, 23]
35
Lista: [35, 14, 23]
Unesite broj koji se trazi: 8
Broj 8 se ne nalazi u listi!
```

(2) U programu se učitani celi brojevi dodaju na kraj liste. Unosi se ceo broj čija se sva pojavljivanja u listi brišu. Na ekran se ispisuje sadržaj liste nakon brisanja.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
2
Lista: [2]
3
Lista: [2, 3]
14
Lista: [2, 3, 14]
3
Lista: [2, 3, 14, 3]
3
Lista: [2, 3, 14, 3, 3]
17
Lista: [2, 3, 14, 3, 3, 17]
3
Lista: [2, 3, 14, 3, 3, 17]
3
Lista: [2, 3, 14, 3, 3, 17, 3]
Unesite broj koji se brise: 3
Lista nakon brisanja: [2, 14, 17]
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
23
Lista: [23]
14
Lista: [23, 14]
35
Lista: [23, 14, 35]
Unesite broj koji se brise: 3
Lista nakon brisanja: [23, 14, 35]
```

(3) U programu se učitani celi brojevi dodaju u listu tako da vrednosti budu uređene u neopadajućem poretku. Unosi se ceo broj koji se traži u unetoj listi i na ekran se ispisuje rezultat pretrage. Potom se unosi još jedan ceo broj čija se sva pojavljivanja u listi brišu i prikazuje se aktuelni sadržaj liste nakon brisanja. Napomena: Prilikom pretraživanja liste i brisanja čvora liste koristiti činjenicu da je lista uređena.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
2
Lista: [2]
3
Lista: [2, 3]
14
Lista: [2, 3, 14]
3
Lista: [2, 3, 3, 14]
5
Lista: [2, 3, 3, 3, 14]
5
Lista: [2, 3, 3, 3, 5, 14]
Unesite broj koji se trazi: 14
Trazeni broj 14 je u listi!
Unesite broj koji se brise: 3
Lista nakon brisanja: [2, 5, 14]
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CTRL+D za kraj unosa):
23
Lista: [23]
14
Lista: [14, 23]
35
Lista: [14, 23, 35]
Unesite broj koji se trazi: 8
Broj 8 se ne nalazi u listi!
Unesite broj koji se brise: 3
Lista nakon brisanja: [14, 23, 35]
```

**Zadatak 4.2** Napisati biblioteku za rad sa jednostruko povezanim listama koja sadrži sve funkcije iz zadatka 4.1, ali tako da funkcije budu implementirane rekurzivno. NAPOMENA: *Koristiti main programe i test primere iz zadatka 4.1.* 

**Zadatak 4.3** Napisati program koji prebrojava pojavljivanja etiketa HTML datoteke čije se ime zadaje kao argument komandne linije. Rezultat prebrojavanja ispisati na standardni izlaz. Etikete smeštati u listu, a za formiranje liste koristiti strukturu:

```
typedef struct _Element
{
   unsigned broj_pojavljivanja;
   char etiketa[20];
   struct _Element *sledeci;
} Element;

   Test 1

| POKRETANJE: ./a.out datoteka.html
| DATOTEKA DATOTEKA.HTML NE POSTOJI.
| IZLAZ ZA GREŠKE:
| Greska: Neuspesno otvaranje
| datoteke datoteka.html.
```

#### Test 2

```
POKRETANJE: ./a.out datoteka.html
                                                   IZLAZ:
                                                     a - 4
                                                     br - 1
DATOTEKA.HTML
                                                     h1 - 2
 <html>
                                                     body - 2
  <head><title>Primer</title></head>
                                                     title - 2
  <body>
                                                     head - 2
    <h1>Naslov</h1>
                                                     html - 2
    Danas je lep i suncan dan. <br>
    A sutra ce biti jos lepsi.
    <a link='http://www.google.com'> Link 1</a>
    <a link='http://www.math.rs'> Link 2</a>
  </body>
 </html>
```

Zadatak 4.4 U datoteci se nalaze podaci o studentima. U svakom redu datoteke nalazi se indeks, ime i prezime studenta. Napisati program kome se preko argumenata komandne linije prosleđuje ime datoteke sa studentskim podacima koje program treba da pročita i smesti u listu. Nakon završenog učitavanja svih podataka o studentima, sa standardnog ulaza unose se, jedan po jedan, indeksi studenata koji se traže u učitanoj listi. Posle svakog unetog indeksa, program ispisuje poruku da ili ne, u zavisnosti od toga da li u listi postoji student sa unetim indeksom ili ne. Prekid unosa indeksa se vrši unošenjem karaktera za kraj ulaza (EOF). Poruke o greškama ispisivati na standardni izlaz za greške. UPUTSTVO: Pretpostaviti da je 10 karaktera dovoljno za zapis indeksa i da je 20 karaktera maksimalna dužina bilo imena bilo prezimena studenta.

```
Primer 1
                                                   Primer 2
POKRETANJE: ./a.out studenti.txt
                                                 POKRETANJE: ./a.out studenti.txt
STUDENTI.TXT
                                                 DATOTEKA STUDENTI.TXT JE PRAZNA
 123/2014 Marko Lukic
 3/2014 Ana Sokic
                                                 INTERAKCIJA SA PROGRAMOM:
 43/2013 Jelena Ilic
                                                   3/2014 ne
 41/2009 Marija Zaric
                                                   235/2008 ne
 13/2010 Milovan Lazic
                                                   41/2009 ne
INTERAKCIJA SA PROGRAMOM:
 3/2014 da: Ana Sokic
 235/2008 ne
 41/2009 da: Marija Zaric
```

- \* Zadatak 4.5 Data je datoteka brojevi.txt koja sadrži cele brojeve.
- (a) Napisati funkciju koja iz zadate datoteke učitava brojeve i smešta ih u listu.
- (b) Napisati funkciju koja u jednom prolazu kroz zadatu listu celih brojeva pronalazi maksimalan strogo rastući podniz.

Napisati program koji u datoteku **rezultat.txt** upisuje nađeni strogo rastući podniz.

```
Test 1

BROJEVI.TXT
43 12 15 16 4 2 8

IZLAZ:
REZULTAT.TXT
12 15 16

Test 2

Test 3

DATOTEKA BROJEVI.TXT | DATOTEKA BROJEVI.TXT JE PRAZNA | DATOTEKA BROJEVI.TXT JE
```

\* Zadatak 4.6 Napisati program koji objedinjuje dve sortirane liste u jednu sortiranu listu. Funkcija ne treba da kreira nove, već da samo preraspodeli postojeće čvorove. Prva lista se učitava iz datoteke čije ime se zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz.

```
Test 1
                                                    Test 2
POKRETANJE: ./a.out dat1.txt dat2.txt
                                                  POKRETANJE: ./a.out dat1.txt dat2.txt
DAT1.TXT
                                                  DAT1.TXT
 2 4 6 10 15
                                                    2 4 6 10 15
DAT2.TXT
                                                  DATOTEKA DAT2.TXT NE POSTOJI.
 5 6 11 12 14 16
                                                  IZLAZ ZA GREŠKE:
IZLAZ:
                                                    Greska: Neuspesno otvaranje
 [2, 4, 5, 6, 6, 10, 11, 12, 14, 15, 16]
                                                   datoteke dat2.txt.
 Test 3
                                                    Test 4
POKRETANJE: ./a.out dat1.txt dat2.txt
                                                 POKRETANJE: ./a.out dat1.txt
DATOTEKA DAT1.TXT JE PRAZNA
                                                  IZLAZ ZA GREŠKE:
                                                    Greska: Program se poziva sa:
DAT2.TXT
                                                   ./a.out dat1.txt dat2.txt!
 5 6 11 12 14 16
TZI.AZ:
 [5, 6, 11, 12, 14, 16]
```

\* Zadatak 4.7 Date su dve jednostruko povezane liste L1 i L2. Napisati funkciju koja od ovih listi formira novu listu L koja sadrži naizmenično raspoređene čvorove listi L1 i L2: prvi čvor iz L1, prvi čvor iz L2, drugi čvor L1, drugi čvor L2, itd. Ne formirati nove čvorove, već samo postojeće rasporediti u jednu listu. Prva lista se učitava iz datoteke čije se ime zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz.

Napomena: Koristiti testove 2 - 6 za zadatak 4.6.

# Test 1 POKRETANJE: ./a.out dat1.txt dat2.txt DAT1.TXT 2 4 6 10 15 DAT2.TXT 5 6 11 12 14 16

2 5 4 6 6 11 10 12 15 14 16

Zadatak 4.8 Sadržaj datoteke je aritmetički izraz koji može sadržati zagrade {, [ i (. Napisati program koji učitava sadržaj datoteke izraz.txt i korišćenjem steka utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz.

```
Test 1
                                                     Test 2
 IZRAZ.TXT
                                                    IZRAZ.TXT
                                                     \{[23 + 5] * (9 * 2)\} - \{23\}
  \{[23 + 5344] * (24 - 234)\} - 23
 IZLAZ:
                                                    IZLAZ:
  Zagrade su ispravno uparene.
                                                    Zagrade su ispravno uparene.
  Test 3
                                                     Test 4
 IZRAZ.TXT
                                                    IZRAZ.TXT
  \{[2 + 54) / (24 * 87)\} + (234 + 23)
                                                     \{(2-14) / (23+11)\}\} * (2+13)
 IZLAZ:
                                                    IZLAZ:
  Zagrade nisu ispravno uparene.
                                                    Zagrade nisu ispravno uparene.
  Test 5
                                                     Test 6
DATOTEKA IZRAZ.TXT JE PRAZNA
                                                   DATOTEKA IZRAZ.TXT NE POSTOJI.
 IZLAZ:
                                                    Izlaz za greške:
 Zagrade su ispravno uparene.
                                                     Greska: Neuspesno otvaranje
                                                     datoteke izraz.txt!
```

Zadatak 4.9 Napisati program koji proverava ispravnost uparivanja etiketa u HTML datoteci. Ime datoteke se zadaje kao argument komandne linije. Poruke o greškama ispisivati na standardni izlaz za greške. UPUTSTVO: Za rešavanje problema koristiti stek implementiran preko liste čiji čvorovi sadrže HTML etikete.

#### Test 1

# Test 2

#### Test 3

#### Test 4

#### Test 5

```
POKRETANJE: ./a.out datoteka.html

DATOTEKA DATOTEKA.HTML NE POSTOJI.

IZLAZ ZA GREŠKE:
Greska: Neuspesno otvaranje
datoteke datoteka.html.
```

Test 6

```
POKRETANJE: ./a.out datoteka.html

DATOTEKA.HTML JE PRAZNA

IZLAZ:
Etikete su pravilno uparene!
```

Zadatak 4.10 Napisati program koji pomaže službeniku u radu na šalteru. Službenik najpre evidentira sve korisničke JMBG brojeve (niske koje sadrže po 13 karaktera) i zahteve (niska koja sadrži najviše 999 karaktera). Prijem zahteva korisnika se prekida unošenjem karaktera za kraj ulaza (EOF). Službenik redom pregleda zahteve i odlučuje da li zahtev obrađuje odmah ili kasnije. Program mu postavlja pitanje Da li korisnika vracate na kraj reda? i ukoliko on da odgovor Da, korisnik se stavlja na kraj reda, čime se obrada njegovog zahteva

odlaže. Ukoliko odgovor nije Da, službenik obrađuje zahtev i podatke o korisniku dopisuje na kraj datoteke izvestaj.txt. Ova datoteka, za svaki obrađen zahtev, sadrži JMBG i zahtev usluženog korisnika. Posle svakog petog usluženog korisnika, službeniku se nudi mogućnost da prekine sa radom, nevezano od broja korisnika koji i dalje čekaju u redu. UPUTSTVO: Za čuvanje korisničkih zahteva koristiti red implementiran korišćenjem listi.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Sluzbenik evidentira korisnicke zahteve:
 Novi zahtev [CTRL+D za kraj]
  JMBG: 1234567890123
  Opis problema: Otvaranje racuna
 Novi zahtev [CTRL+D za kraj]
  JMBG: 2345678901234
  Opis problema: Podizanje novca
 Novi zahtev [CTRL+D za kraj]
  JMBG: 3456789012345
  Opis problema: Reklamacija
 Novi zahtev [CTRL+D za kraj]
  .IMRG:
 Sledeci je korisnik sa JMBG: 1234567890123
 i zahtevom: Otvaranje racuna
  Da li ga vracate na kraj reda? [Da/Ne] Da
 Sledeci je korisnik sa JMBG: 2345678901234
 i zahtevom: Podizanje novca
  Da li ga vracate na kraj reda? [Da/Ne] Ne
 Sledeci je korisnik sa JMBG: 3456789012345
 i zahtevom: Reklamacija
  Da li ga vracate na kraj reda? [Da/Ne] Da
 Sledeci je korisnik sa JMBG: 1234567890123
 i zahtevom: Otvaranje racuna
  Da li ga vracate na kraj reda? [Da/Ne] Da
 Sledeci je korisnik sa JMBG: 3456789012345
 i zahtevom: Reklamacija
  Da li ga vracate na kraj reda? [Da/Ne] Ne
 Da li je kraj smene? [Da/Ne] Ne
 Sledeci je korisnik sa JMBG: 1234567890123
 i zahtevom: Otvaranje racuna
  Da li ga vracate na kraj reda? [Da/Ne] Ne
IZVESTAJ.TXT
  JMBG: 2345678901234 Zahtev: Podizanje novca
  JMBG: 3456789012345 Zahtev: Reklamacija
  JMBG: 1234567890123 Zahtev: Otvaranje racuna
```

Zadatak 4.11 Napisati biblioteku za rad sa dvostruko povezanom listom celih brojeva koja ima iste funkcionalnosti kao biblioteka iz zadatka 4.1. Dopuniti biblioteku novim funkcijama.

- (a) Napisati funkciju void obrisi\_tekuci(Cvor \*\* adresa\_glave, Cvor \*\* adresa\_kraja, Cvor \* tekuci) koja briše čvor na koji pokazuje pokazivač tekuci iz liste čiji se pokazivač na čvor koji je glava liste nalazi na adresi adresa\_glave i poslednji čvor liste na adresi adresa\_kraja.
- (b) Napisati funkciju void ispisi\_listu\_unazad(Cvor \* kraj) koja ispisuje sadržaj liste od poslednjeg čvora ka glavi liste.

Sve funkcije za rad sa listom implementirati iterativno. Zbog efikasnog izvršavanja operacija dodavanja na kraj liste i ispisivanja liste unazad treba, pored pokazivača na glavu liste, čuvati i pokazivač na poslednji čvor liste. NAPOMENA: Koristiti test primere iz zadatka 4.1

\* Zadatak 4.12 Grupa od n plesača na kostimima ima brojeve od 1 do n. Plesači najpre formiraju krug tako da brojevi sa njihovih kostima rastu u smeru kazaljke na satu. Plesač sa brojem 1 stavlja levu ruku na rame plesača sa brojem 2, a desnu na svoj kuk i tako redom. Plesač sa brojem n svoju levu ruku spušta na rame plesača sa brojem 1, a desnu na svoj kuk i tako zatvara krug. Svoju plesnu tačku izvode tako što iz formiranog kruga najpre izlazi k-ti plesač. Odbrojava se počevši od plesača označenog brojem 1 u smeru kretanja kazaljke na satu. Preostali plesači obrazuju manji krug tako što k-1-vi stavlja ruku na rame k+1-og i zatvara krug iz kog opet izlazi k-ti plesač. Odbrojavanje sada počinje od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu. Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi n, k (k < n) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga. UPUTSTVO: Pri implementaciji koristiti jednostruko povezanu kružnu listu.

Test 1	Test 2	Test 3
ULAZ:   5 3	ULAZ:    8 4	ULAZ:   3 8
IzLAZ: 3 1 5 2 4	IZLAZ: 4 8 5 2 1 3 7 6	IZLAZ ZA GREŠKE: Greska: n mora biti uvek vece od k, a 3 < 8!

 ${\bf *}$  Zadatak 4.13 Grupa od n plesača na kostimima ima brojeve od 1 do n. Plesači najpre formiraju krug tako da brojevi sa njihovih kostima rastu u smeru kazaljke na satu. Svaki plesač levu ruku stavlja na rame plesača sa sledećim većim

brojem, a desnu na rame plesača sa prvim manjim brojem. Plesač sa brojem 1 stavlja levu ruku na rame plesača sa brojem 2, a desnu na rame plesača sa brojem n. Plesač sa brojem n svoju desnu ruku spušta na rame plesača sa brojem n-1, a levu na rame plesača sa brojem 1 i tako zatvara krug. Plesači izvode svoju plesnu tačku tako što iz formiranog kruga najpre izlazi k-ti plesač. Odbrojava se počevši od plesača označenog brojem 1 u smeru kretanja kazaljke na satu. Preostali plesači obrazuju manji krug iz kog opet izlazi k-ti plesač. Odbrojavanje sada počinje od sledećeg suseda prethodno izbačenog, uz promenu smera. Ukoliko se prilikom prethodnog izbacivanja odbrojavalo u smeru kazaljke na satu sada će se obrojavati u suprotnom smeru, i obrnuto. Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi n, k (k < n) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga. UPUTSTVO: Pri implementaciji koristiti dvostruko povezanu kružnu listu.

## 4.2 Stabla

Zadatak 4.14 Napisati biblioteku za rad sa binarnim pretraživačkim stablima.

- (a) Definisati strukturu Cvor kojom se opisuje čvor stabla, a koja sadrži ceo broj broj i pokazivače levo i desno redom na levo i desno podstablo.
- (b) Napisati funkciju Cvor \*napravi\_cvor(int broj) koja alocira memoriju za novi čvor stabla i vrši njegovu inicijalizaciju zadatim celim brojem broj.
- (c) Napisati funkciju int dodaj\_u\_stablo(Cvor \*\* adresa\_korena, int broj) koja u stablo na koje pokazuje argument adresa\_korena dodaje ceo broj broj. Povratna vrednost funkcije je 0 ako je dodavanje uspešno, odnosno 1 ukoliko je došlo do greške.
- (d) Napisati funkciju Cvor \*pretrazi\_stablo(Cvor \* koren, int broj) koja proverava da li se ceo broj broj nalazi u stablu sa korenom koren. Funk-

- cija vraća pokazivač na čvor stabla koji sadrži traženu vrednost ili NULL ukoliko takav čvor ne postoji.
- (e) Napisati funkciju Cvor \*pronadji\_najmanji(Cvor \* koren) koja pronalazi čvor koji sadrži najmanju vrednost u stablu sa korenom koren.
- (f) Napisati funkciju Cvor \*pronadji\_najveci(Cvor \* koren) koja pronalazi čvor koji sadrži najveću vrednost u stablu sa korenom koren.
- (g) Napisati funkciju void obrisi\_element(Cvor \*\* adresa\_korena, int broj) koja briše čvor koji sadrži vrednost broj iz stabla na koje pokazuje argument adresa\_korena.
- (h) Napisati funkciju void ispisi\_stablo\_infiksno(Cvor \* koren) koja infiksno ispisuje sadržaj stabla sa korenom koren. Infiksni ispis podrazumeva ispis levog podstabla, korena, a zatim i desnog podstabla.
- (i) Napisati funkciju void ispisi\_stablo\_prefiksno(Cvor \* koren) koja prefiksno ispisuje sadržaj stabla sa korenom koren. Prefiksni ispis podrazumeva ispis korena, levog podstabla, a zatim i desnog podstabla.
- (j) Napisati funkciju void ispisi\_stablo\_postfiksno(Cvor \* koren) koja postfiksno ispisuje sadržaj stabla sa korenom koren. Postfiksni ispis podrazumeva ispis levog podstabla, desnog podstabla, a zatim i korena.
- (k) Napisati funkciju void oslobodi\_stablo(Cvor \*\* adresa\_korena) koja oslobađa memoriju zauzetu stablom na koje pokazuje argument adresa\_korena.

Korišćenjem kreirane biblioteke, napisati program koji sa standardnog ulaza učitava cele brojeve sve do kraja ulaza, dodaje ih u binarno pretraživačko stablo i ispisuje stablo u svakoj od navedenih notacija. Zatim omogućiti unos još dva cela broja i demonstrirati rad funkcije za pretragu nad prvim unetim brojem i rad funkcije za brisanje elemenata nad drugim unetim brojem.

#### Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite brojeve (CRL+D za kraj unosa):
7 2 1 9 32 18
Infiksni ispis: 1 2 7 9 18 32
Prefiksni ispis: 7 2 1 9 32 18
Postfiksni ispis: 1 2 18 32 9 7
Trazi se broj: 11
Broj se ne nalazi u stablu!
Brise se broj: 7
Rezultujuce stablo: 1 2 9 18 32
```

#### Primer 2

```
| INTERAKCIJA SA PROGRAMOM:
| Unesite brojeve (CRL+D za kraj unosa):
| 8 -2 6 13 24 -3 |
| Infiksni ispis: -3 -2 6 8 13 24 |
| Prefiksni ispis: 8 -2 -3 6 13 24 |
| Postfiksni ispis: -3 6 -2 24 13 8 |
| Trazi se broj: 6 |
| Broj se nalazi u stablu! |
| Brise se broj: 14 |
| Rezultujuce stablo: -3 -2 6 8 13 24
```

Zadatak 4.15 Napisati program koji izračunava i na standardni izlaz ispisuje broj pojavljivanja svake reči datoteke čije se ime zadaje kao argument komandne linije. Program realizovati korišćenjem binarnog pretraživačkog stabla uređenog leksikografski po rečima ne uzimajući u obzir razliku između malih i velikih slova. Ukoliko prilikom pokretanja programa korisnik ne navede ime ulazne datoteke ispisati na standardni izlaz za grešku poruku Nedostaje ime ulazne datoteke!. Može se pretpostaviti da dužina reči neće biti veća od 50 karaktera.

Test 2 Test 1 POKRETANJE: ./a.out test.txt POKRETANJE: ./a.out suma.txt TEST. TXT SIIMA TXT Sunce utorak raCunar SUNCE programiranje lipa zova hrast ZOVA breza LIPA jabuka PROGramiranje sunCE JABUka IZLAZ: IZLAZ: breza: 1 jabuka: 2 hrast: 1 programiranje: 2 lipa: 2 racunar: 1 zova: 2 sunce: 3 utorak: 1 Najcesca rec: lipa (pojavljuje se 2 puta) Najcesca rec: sunce (pojavljuje se 3 puta)

```
Test 3

POKRETANJE: ./a.out ulaz.txt

DATOTEKA ULAZ.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
Greska: Nedostaje ime ulazne datoteke!

Greska: Neuspesno otvaranje datoteke ulaz.txt.
```

Zadatak 4.16 U svakoj liniji datoteke čije se ime zadaje sa standardnog ulaza nalazi se ime osobe, prezime osobe i njen broj telefona međusobno razdvojeni blanko znakom, na primer  $Pera\ Peric\ 064/123-4567$ . Napisati program koji korišćenjem binarnog pretraživačkog stabla implementira mapu koja sadrži navedene informacije i koja će omogućiti pretragu brojeva telefona za zadata imena i prezimena. Imena i prezimena se unose sve do unosa reči KRAJ, a za svaki od unetih podataka ispisuje se ili broj telefona ili obaveštenje da traženi broj nije u imeniku. Može se pretpostaviti da imena, prezimena i brojevi telefona neće biti duži od 30 karaktera, kao i da imenik ne sadrži podatke o osobama sa istim imenom i prezimenom.

#### Primer 1

```
IMENIK.TXT

Pera Peric 011/3240-987

Marko Maric 064/1234-987

Mirko Maric 011/589-333

Sanja Savkovic 063/321-098

Zika Zikic 021/759-858

INTERAKCIJA SA PROGRAMOM:

Unesite ime datoteke: imenik.txt

Unesite ime i prezime: Pera Peric

Broj je: 011/3240-987

Unesite ime i prezime: Marko Markovic

Broj nije u imeniku!

Unesite ime i prezime: KRAJ
```

#### Primer 2

```
| DATOTEKA IMENIK1.TXT NE POSTOJI

INTERAKCIJA SA PROGRAMOM:
Unesite ime datoteke: imenik1.txt

IZLAZ ZA GREŠKE:
Greska: Neuspesno otvaranje datoteke
imenik1.txt.
```

Zadatak 4.17 U datoteci prijemni.txt nalaze se podaci o prijemnom ispitu učenika jedne osnovne škole tako što je u svakom redu navedeno ime i prezime učenika (niska od najviše 50 karaktera), broj poena na osnovu uspeha (realan broj), broj poena na prijemnom ispitu iz matematike (realan broj) i broj poena na prijemnom ispitu iz maternjeg jezika (realan broj). Za učenika koji u zbiru osvoji manje od 10 poena na oba prijemna ispita smatra se da nije položio prijemni. Napisati program koji na osnovu podataka iz ove datoteke formira i prikazuje rang listu učenika. Rang lista sadrži redni broj učenika, njegovo ime i prezime, broj poena na osnovu uspeha, broj poena na prijemnom ispitu iz maternjeg jezika i ukupan broj poena i sortirana je opadajuće po ukupnom broju poena. Na rang listi se prvo navode oni učenici koji su položili prijemni ispit, a potom i učenici koji ga nisu položili. Između ovih dveju grupa učenika postoji i horizontalna linija koja ih vizuelno razdvaja.

#### Test 1

```
PRIJEMNI.TXT

Marko Markovic 45.4 12.3 11

Milan Jevremovic 35.2 1.3 9

Maja Agic 60 19 20

Nadica Zec 54.2 10 15.8

Jovana Milic 23.3 2 5.6

IZLAZ:

1. Maja Agic 60.0 19.0 20.0 99.0

2. Nadica Zec 54.2 10.0 15.8 80.0

3. Marko Markovic 45.4 12.3 11.0 68.7

4. Milan Jevremovic 35.2 1.3 9.0 45.5

5. Jovana Milic 23.3 2.0 5.6 30.9
```

#### Test 2

```
DATOTEKA PRIJEMNI.TXT NE POSTOJI

IZLAZ ZA GREŠKE:

Greska: Neuspesno otvaranje datoteke
prijemni.txt.
```

\* Zadatak 4.18 Napisati program koji implementira podsetnik za rođendane. Informacije o rođendanima se nalaze u datoteci čije se ime zadaje kao argument komandne linije. Svaka linija datoteke je formata Ime Prezime DD.MM. i sadži ime osobe, prezime osobe i dan i mesec rođenja. Korisnik unosi datum u naznačenom formatu, a program pronalazi i ispisuje ime i prezime osobe čiji je rođendan zadatog datuma ili ime i prezime osobe koja prva sledeća slavi rođendan. Ovaj postupak treba ponavljati dokle god korisnik ne unese komandu za kraj unosa. Informacije o rođendanima uneti u mapu koja je implementirana preko binarnog pretraživačkog stabla i uređena po datumima - prvo po mesecu, a zatim po danu u okviru istog meseca. Može se pretpostaviti da će svi korišćeni datumi biti validni i DD.MM. formata. Takođe, može se pretpostaviti da će ime i prezime osobe biti kraće od 50 karaktera.

#### Primer 1

```
POKRETANJE: ./a.out rodjendani.txt
RODJENDANI, TXT
 Marko Markovic 12.12.
 Milan Jevremovic 04.06.
 Maja Agic 23.04.
 Nadica Zec 01.01.
 Jovana Milic 05.05.
INTERAKCIJA SA PROGRAMOM:
 Unesite datum: 23.04.
 Slavljenik: Maja Agic
 Unesite datum: 01.01.
 Slavljenik: Nadica Zec
 Unesite datum: 01.05.
 Slavljenik: Jovana Milic 05.05.
 Unesite datum: 20.12.
 Slavljenik: Nadica Zec 01.01.
 Unesite datum:
```

#### Primer 2

```
POKRETANJE: ./a.out rodjendani.txt

DATOTEKA RODJENDANI.TXT NE POSTOJI

IZLAZ ZA GREŠKE:
Greska: Neuspesno otvaranje datoteke rodjendani.txt.
```

Zadatak 4.19 Dva binarna stabla su identična ako su ista po strukturi i sadržaju tj. ako oba korena imaju isti sadržaj i identična odgovarajuća podstabla. Napistati funkciju int identitet(Cvor \* koren1, Cvor \* koren2) koja proverava da li su binarna stabla koren1 i koren2 koja sadrže cele brojeve identična, a zatim i glavni program koji testira njen rad. Elemente pojedinačnih stabla unositi sa standardnog ulaza sve do pojave broja 0. NAPOMENA: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

#### Primer 1

```
| INTERAKCIJA SA PROGRAMOM:
| Prvo stablo:
| 10 5 15 3 2 4 30 12 14 13 0
| Drugo stablo:
| 10 15 5 3 4 2 12 14 13 30 0
| Stabla jesu identicna.
```

#### Primer 2

```
| INTERAKCIJA SA PROGRAMOM:

| Prvo stablo:
| 10 5 15 4 3 2 30 12 14 13 0
| Drugo stablo:
| 10 15 5 3 4 2 12 14 13 30 0
| Stabla nisu identicna.
```

\* Zadatak 4.20 Napisati program za rad sa skupovima u kojem se skupovi predstavljaju pomoću binarnih pretraživačkih stabala. Program za dva skupa čiji se elementi zadaju sa standardnog ulaza, sve do kraja ulaza, ispisuje uniju, presek i razliku skupova. NAPOMENA: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

#### Primer 1

```
| INTERAKCIJA SA PROGRAMOM:
| Prvi skup: 1 7 8 9 2 2
| Drugi skup: 3 9 6 11 1
| Unija: 1 1 2 2 3 6 7 8 9 9 11
| Presek: 1 9
| Razlika: 2 2 7 8
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Prvi skup: 11 2 7 5
Drugi skup: 4 3 3 7
Unija: 2 3 3 4 5 7 7 11
Presek: 7
Razlika: 2 5 11
```

Zadatak 4.21 Napisati funkciju void sortiraj (int a[], int n) koja sortira niz celih brojeva a dimenzije n korišćenjem binarnog pretraživačkog stabla. Napisati i program koji sa standardnog ulaza učitava ceo broj n manji od 50 i niz a celih brojeva dužine n, poziva funkciju sortiraj i rezultat ispisuje na standardni izlaz. Napomena: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

#### Primer 1

```
| INTERAKCIJA SA PROGRAMOM:

n: 7

a: 1 11 8 6 37 25 30

1 6 8 11 25 30 37
```

#### Primer 2

```
INTERAKCIJA SA PROGRAMOM:

n: 55

Greska: Pogresna dimenzija niza!
```

Zadatak 4.22 Dato je binarno pretraživačko stablo celih brojeva.

- (a) Napisati funkciju koja izračunava broj čvorova stabla.
- (b) Napisati funkciju koja izračunava broj listova stabla.
- (c) Napisati funkciju koja štampa pozitivne vrednosti listova stabla.

- (d) Napisati funkciju koja izračunava zbir čvorova stabla.
- (e) Napisati funkciju koja izračunava najveći element stabla.
- (f) Napisati funkciju koja izračunava dubinu stabla.
- (g) Napisati funkciju koja izračunava broj čvorova na *i*-tom nivou stabla.
- (h) Napisati funkciju koja ispisuje sve elemente na *i*-tom nivou stabla.
- Napisati funkciju koja izračunava maksimalnu vrednost na i-tom nivou stabla.
- (j) Napisati funkciju koja izračunava zbir čvorova na *i*-tom nivou stabla.
- (k) Napisati funkciju koja izračunava zbir svih vrednosti stabla koje su manje ili jednake od date vrednosti x.

Napisati program koji testira prethodne funkcije. Stablo formirati na osnovu vrednosti koje se unose sa standardnog ulaza, sve do kraja ulaza, a vrednosti parametara i i x pročitati kao argumente komandne linije. NAPOMENA: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

```
Test 1
                                                     Test 2
                                                   POKRETANJE: ./a.out 3 31
POKRETANJE: ./a.out 2 15
ULAZ:
                                                   ULAZ:
                                                     24 53 61 9 7 55 20 16
 10 5 15 3 2 4 30 12 14 13
TZI.AZ:
                                                   TZI.AZ:
 Broj cvorova: 10
                                                     Broj cvorova: 8
 Broj listova: 4
                                                     Broj listova: 3
 Pozitivni listovi: 2 4 13 30
                                                     Pozitivni listovi: 7 16 55
 Zbir cvorova: 108
                                                     Zbir cvorova: 245
 Naiveci element: 30
                                                     Naiveci element: 61
 Dubina stabla: 5
                                                     Dubina stabla: 4
 Broj cvorova na 2. nivou: 3
                                                     Broj cvorova na 3. nivou: 2
 Elementi na 2. nivou: 3 12 30
                                                     Elementi na 3. nivou: 16 55
 Maksimalni element na 2. nivou: 30
                                                     Maksimalni element na 3. nivou: 55
 Zbir elemenata na 2. nivou: 45
                                                     Zbir elemenata na 3. nivou: 71
 Zbir elemenata manjih ili jednakih od 15:
                                                     Zbir elemenata manjih ili jednakih od 31:
```

Zadatak 4.23 Napisati program koji ispisuje sadržaj binarnog pretraživačkog stabla po nivoima. Elementi stabla se učitavaju sa standardnog ulaza sve do kraja ulaza. Napomena: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

```
Test 3
Test 1
                               Test 2
ULAZ:
                                                            ULAZ:
 10 5 15 3 2 4 30 12 14 13
                                6 11 8 3 -2
                                                               24 53 61 9 7 55 20 16
                                                             IZLAZ:
 0.nivo: 10
                                0.nivo: 6
                                                               0.nivo: 24
 1.nivo: 5 15
                                1.nivo: 3 11
                                                               1.nivo: 9 53
 2.nivo: 3 12 30
                                2.nivo: -2 8
                                                               2.nivo: 7 20 61
 3.nivo: 2 4 14
                                                               3.nivo: 16 55
 4.nivo: 13
```

\* Zadatak 4.24 Dva binarna stabla su slična kao u ogledalu ako su ili oba prazna ili ako oba nisu prazna i levo podstablo svakog stabla je slično kao u ogledalu desnom podstablu onog drugog (bitna je struktura stabala, ali ne i njihov sadržaj). Napisati funkciju koja proverava da li su dva binarna pretraživačka stabla slična kao u ogledalu, a potom i program koji testira rad funkcije nad stablima čiji se elementi unose sa standardnog ulaza sve do unosa broja 0 i to redom za prvo stablo, pa zatim i za drugo stablo.

```
Primer 1

| Interakcija sa programom: | Interakcija sa programom: | Prvo stablo: 11 20 5 3 0 | Prvo stablo: 8 14 30 1 0 | Prugo stablo: 8 20 15 0 | Stabla su slicna kao u ogledalu. | Stabla nisu slicna kao u ogledalu.
```

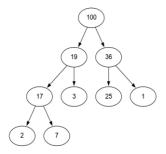
Zadatak 4.25 AVL-stablo je binarno pretraživačko stablo kod kojeg apsolutna razlika visina levog i desnog podstabla svakog elementa nije veća od jedan. Napisati funkciju int avl(Cvor \* koren) koja izračunava broj čvorova stabla sa korenom koren koji ispunjavaju uslov za AVL stablo. Napisati zatim i glavni program koji ispisuje rezultat avl funkcije za stablo čiji se elementi unose sa standardnog ulaza sve do kraja ulaza. Napomena: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

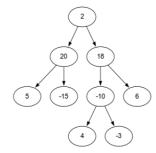
Zadatak 4.26 Binarno stablo celih pozitivnih brojeva se naziva *hip* (engl. *heap*) ako za svaki čvor u stablu važi da je njegova vrednost veća od vrednosti svih ostalih čvorova u njegovim podstablima. Napisati funkciju int hip(Cvor \* koren)

koja proverava da li je dato binarno stablo celih brojeva hip. Napisati zatim i glavni program koji kreira stablo zadato slikom 4.1, poziva funkciju hip i ispisuje rezultat na standardni izlaz. Napomena: Za alokaciju i oslobađanje memorije koristiti funkcije napravi\_cvor i oslobodi\_stablo iz zadatka 4.14.

#### Test 1

```
IZLAZ:
Zadato stablo je hip!
```





Slika 4.1: Zadatak 4.26

Slika 4.2: Zadatak 4.27

#### Zadatak 4.27 Dato je binarno stablo celih brojeva.

- (a) Napisati funkciju koja pronalazi čvor u stablu sa najvećim zbirom vrednosti iz desnog podstabla.
- (b) Napisati funkciju koja pronalazi čvor u stablu sa najmanjim zbirom vrednosti iz levog podstabla.
- (c) Napisati funkciju koja štampa sadržaj svih čvorova stabla na putanji od korena do najdubljeg čvora.
- (d) Napisati funkciju koja štampa sadržaj svih čvorova stabla na putanji od korena do čvora koji ima najmanju vrednost u stablu.

Napisati program koji testira gore navedene funkcije nad stablom zadatim slikom 4.2 i rezultat ispisuje na standardni izlaz.

#### Test 1

```
| IZLAZ:
| Vrednost u cvoru sa maksimalnim desnim zbirom: 18
| Vrednost u cvoru sa minimalnim levim zbirom: 18
| 2 18 -10 4
| 2 20 -15
```

# 4.3 Rešenja

#### Rešenje 4.1

#### lista.h

```
#ifndef _LISTA_H_
2 #define _LISTA_H_
 //* Struktura kojom je predstavljen cvor liste sadrzi celobrojni
     podatak vrednost i pokazivac na sledeci cvor liste */
6 typedef struct cvor {
   int vrednost;
   struct cvor *sledeci;
  } Cvor;
  /* Funkcija kreira cvor, vrednost novog cvora inicijalizuje na broj,
    dok pokazivac na sledeci cvor postavlja na NULL. Vraca pokazivac
     na novokreirani cvor ili NULL ako alokacija nije bila uspesna. */
14 Cvor *napravi_cvor(int broj);
16 /* Funkcija oslobadja dinamicku memoriju zauzetu za cvorove liste
     ciji se pokazivac glava nalazi na adresi adresa_glave. */
void oslobodi_listu(Cvor ** adresa_glave);
20 /* Funkcija dodaje broj na pocetak liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije, inace vraca 0. */
22 int dodaj_na_pocetak_liste(Cvor ** adresa_glave, int broj);
24 /* Funkcija pronalazi i vraca pokazivac na poslednji cvor liste, ili
     NULL ukoliko je lista prazna. */
26 Cvor *pronadji_poslednji(Cvor * glava);
28 /* Funkcija dodaje broj na kraj liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije, inace vraca 0. */
30 int dodaj_na_kraj_liste(Cvor ** adresa_glave, int broj);
32 /* Pomocna funkcija pronalazi cvor u listi iza koga treba umetnuti
    nov cvor sa vrednoscu broj. */
34 Cvor *pronadji_mesto_umetanja(Cvor * glava, int broj);
36 /* Funkcija dodaje broj iza zadatog cvora. Vraca 1 ukoliko je
     dodavanje uspesno, odnosno 0 ukoliko je doslo do greske. */
38 int dodaj_iza(Cvor * tekuci, int broj);
40 /* Funkcija dodaje broj u sortiranu listu tako da lista ostane
    sortirana. Vraca 1 ukoliko je bilo greske pri alokaciji memorije,
    inace vraca 0. */
  int dodaj_sortirano(Cvor ** adresa_glave, int broj);
44
```

```
/* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
     Vraca pokazivac na cvor liste u kome je sadrzan trazeni broj ili
     NULL u slucaju da takav cvor ne postoji u listi. */
48 Cvor *pretrazi_listu(Cvor * glava, int broj);
  /* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
     U pretrazi oslanja se na cinjenicu da je lista koja se pretrazuje
     neopadajuce sortirana. Vraca pokazivac na cvor liste u kome je
     sadrzan trazeni broj ili NULL u slucaju da takav cvor ne postoji.
      */
54 Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj);
  /* Funkcija brise iz liste sve cvorove koji sadrze dati broj.
     Azurira pokazivac na glavu liste, koji moze biti promenjen u
     slucaju da se obrise stara glava. */
  void obrisi_cvor(Cvor ** adresa_glave, int broj);
60
  /* Funkcija brise iz liste sve cvorove koji sadrze dati broj,
     oslanjajuci se na cinjenicu da je prosledjena lista sortirana
62
     neopadajuce. Azurira pokazivac na glavu liste, koji moze biti
     promenjen ukoliko se obrise stara glava liste. */
64
  void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, int broj);
66
  /* Funkcija prikazuje vrednosti cvorova liste pocev od glave ka
     kraju liste, razdvojene zapetama i uokvirene zagradama. */
68
  void ispisi_listu(Cvor * glava);
  #endif
```

#### lista.c

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include "lista.h"
5 Cvor *napravi_cvor(int broj)
    /* Alokacija memorije za novi cvor uz proveru uspesnosti */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
      return NULL;
    /* Inicijalizacija polja strukture */
    novi->vrednost = broj;
    novi->sledeci = NULL;
    /* Vraca se adresa novog cvora */
    return novi;
17
  }
19
  void oslobodi_listu(Cvor ** adresa_glave)
```

```
21 {
    Cvor *pomocni = NULL;
    /* Ako lista nije prazna, onda treba osloboditi memoriju */
    while (*adresa_glave != NULL) {
      /* Potrebno je prvo zapamtiti adresu sledeceg cvora i onda
         osloboditi cvor koji predstavlja glavu liste */
      pomocni = (*adresa_glave)->sledeci;
      free(*adresa_glave);
      /* Sledeci cvor je nova glava liste */
      *adresa_glave = pomocni;
  }
35
  int dodaj_na_pocetak_liste(Cvor ** adresa_glave, int broj)
37
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
      return 1;
41
    /* Novi cvor se uvezuje na pocetak i postaje nova glave liste */
43
    novi->sledeci = *adresa_glave;
    *adresa_glave = novi;
45
    /* Vraca se indikator uspesnog dodavanja */
    return 0;
  }
49
  Cvor *pronadji_poslednji(Cvor * glava)
    /* U praznoj listi nema cvorova pa se vraca NULL */
53
    if (glava == NULL)
      return NULL;
    /* Sve dok glava pokazuje na cvor koji ima sledbenika, pokazivac
       glava se pomera na sledeci cvor. Nakon izlaska iz petlje, glava
       ce pokazivati na cvor liste koji nema sledbenika, tj. na
59
       poslednji cvor liste pa se vraca vrednost pokazivaca glava.
       Pokazivac glava je argument funkcije i njegove promene nece se
       odraziti na vrednost pokazivaca glava u pozivajucoj funkciji. */
    while (glava->sledeci != NULL)
      glava = glava->sledeci;
    return glava;
67
69 int dodaj_na_kraj_liste(Cvor ** adresa_glave, int broj)
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
    Cvor *novi = napravi_cvor(broj);
```

```
73
     if (novi == NULL)
       return 1;
     /* Ako je lista prazna */
     if (*adresa_glave == NULL) {
77
       /* Glava nove liste je upravo novi cvor */
       *adresa_glave = novi;
79
     } else {
       /* Ako lista nije prazna, pronalazi se poslednji cvor i novi
81
          cvor se dodaje na kraj liste kao sledbenik poslednjeg */
       Cvor *poslednji = pronadji_poslednji(*adresa_glave);
83
       poslednji->sledeci = novi;
85
     /* Vraca se indikator uspesnog dodavanja */
     return 0;
   }
89
   Cvor *pronadji_mesto_umetanja(Cvor * glava, int broj)
91
     /* U praznoj listi nema takvog mesta i vraca se NULL */
93
     if (glava == NULL)
       return NULL;
95
     /* Pokazivac glava se pomera na sledeci cvor sve dok ne bude
97
        pokazivao na cvor ciji sledeci ili ne postoji ili ima vrednost
        vecu ili jednaku vrednosti novog cvora.*/
99
     /* Zbog izracunavanja izraza u C-u prvi deo konjunkcije mora biti
        provera da li se doslo do poslednjeg cvora liste pre nego sto se
        proveri vrednost u sledecem cvoru, jer u slucaju poslednjeg,
        sledeci ne postoji, pa ni njegova vrednost. */
     while (glava->sledeci != NULL && glava->sledeci->vrednost < broj)
       glava = glava->sledeci;
     /* Iz petlje se moglo izaci pomeranjem pokazivaca glava do
        poslednjeg cvora ili, ranije, nailaskom na cvor ciji sledeci
        ima vrednost vecu od broj. */
     return glava;
   }
   int dodaj_iza(Cvor * tekuci, int broj)
     /* Kreira se novi cvor i proverava se uspesnost kreiranja */
     Cvor *novi = napravi_cvor(broj);
     if (novi == NULL)
117
       return 1;
119
     /* Novi cvor se dodaje iza tekuceg cvora. */
     novi->sledeci = tekuci->sledeci;
     tekuci->sledeci = novi;
123
     /* Vraca se indikator uspesnog dodavanja */
```

```
return 0;
   int dodaj_sortirano(Cvor ** adresa_glave, int broj)
129 {
     /* Ako je lista prazna */
    if (*adresa_glave == NULL) {
      /* Glava nove liste je novi cvor */
       /* Kreiranje novog cvora uz proveru uspesnost kreiranja */
       Cvor *novi = napravi_cvor(broj);
       if (novi == NULL)
        return 1;
       *adresa_glave = novi;
139
       /* Vraca se indikator uspesnog dodavanja */
      return 0;
141
143
     /* Inace, ako je broj manji ili jednak vrednosti u glavi liste,
       onda ga treba dodati na pocetak liste. */
     if ((*adresa_glave)->vrednost >= broj) {
      return dodaj_na_pocetak_liste(adresa_glave, broj);
147
149
     /* U slucaju da je glava liste cvor sa vrednoscu manjom od broj,
       tada se pronalazi cvor liste iza koga treba uvezati nov cvor */
     Cvor *pomocni = pronadji_mesto_umetanja(*adresa_glave, broj);
     return dodaj_iza(pomocni, broj);
   Cvor *pretrazi_listu(Cvor * glava, int broj)
157 {
     /* Obilaze se cvorovi liste */
    for (; glava != NULL; glava = glava->sledeci)
159
       /* Ako je vrednost tekuceg cvora jednaka zadatom broju, pretraga
161
         se obustavlja */
       if (glava->vrednost == broj)
        return glava;
    /* Nema trazenog broja u listi i vraca se NULL */
    return NULL;
167 }
Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj)
     /* Obilaze se cvorovi liste */
    /* U uslovu ostanka u petlji, bitan je redosled provera u
       konjunkciji. */
    while (glava != NULL && glava->vrednost < broj)
       glava = glava->sledeci;
```

```
/* Iz petlje se moglo izaci na vise nacina. Prvi, tako sto je
        glava->vrednost veca od trazenog broja i tada treba vratiti
        NULL, jer trazen broj nije nadjen medju manjim brojevima pri
        pocetku sortirane liste, pa se moze zakljuciti da ga nema u
        listi. Drugi nacini, tako sto se doslo do kraja liste i glava
181
        je NULL ili tako sto je glava->vrednost == broj. U oba
        poslednja nacina treba vratiti pokazivac glava bilo da je NULL
183
        ili pokazivac na konkretan cvor. */
     if (glava->vrednost > broj)
185
       return NULL;
     else
187
       return glava;
   }
189
   void obrisi_cvor(Cvor ** adresa_glave, int broj)
191
     Cvor *tekuci = NULL;
193
     Cvor *pomocni = NULL;
     /* Sa pocetka liste se brisu svi cvorovi koji su jednaki datom
        broju i azurira se pokazivac na glavu liste */
     while (*adresa_glave != NULL && (*adresa_glave)->vrednost == broj)
       /* Adresu repa liste treba sacuvati pre oslobadjanja cvora na
190
          adresi adresa_glave */
       pomocni = (*adresa_glave)->sledeci;
201
       free(*adresa_glave);
       *adresa_glave = pomocni;
203
205
     /* Ako je nakon ovog brisanja lista ostala prazna, izlazi se iz
207
        funkcije */
     if (*adresa_glave == NULL)
       return;
209
     /* Od ovog trenutka, u svakoj iteraciji petlje promenljiva tekuci
        pokazuje na cvor cija je vrednost razlicita od trazenog broja.
        Isto vazi i za sve cvorove levo od tekuceg. Poredi se vrednost
        sledeceg cvora (ako postoji) sa trazenim brojem. Cvor se brise
        ako je jednak, a ako je razlicit, prelazi se na sledeci cvor.
215
        Ovaj postupak se ponavlja dok se ne dodje do poslednjeg cvora.
     tekuci = *adresa_glave;
217
     while (tekuci->sledeci != NULL)
       if (tekuci->sledeci->vrednost == broj) {
219
         /* tekuci->sledeci treba obrisati, zbog toga se njegova adresa
            prvo cuva u promenljivoj pomocni. */
221
         pomocni = tekuci->sledeci;
         /* Tekucem se preusmerava pokazivac sledeci, preskakanjem
            njegovog trenutnog sledeceg. Njegov novi sledeci ce biti
            sledeci od cvora koji se brise. */
         tekuci->sledeci = pomocni->sledeci;
```

```
/* Sada treba osloboditi cvor sa vrednoscu broj. */
         free(pomocni);
       } else {
         /* Inace, ne treba brisati sledeceg od tekuceg i pokazivac se
            pomera na sledeci. */
231
         tekuci = tekuci->sledeci;
     return:
235 }
void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, int broj)
     Cvor *tekuci = NULL:
     Cvor *pomocni = NULL;
     /* Sa pocetka liste se brisu svi cvorovi koji su jednaki datom
        broju i azurira se pokazivac na glavu liste. */
     while (*adresa_glave != NULL && (*adresa_glave)->vrednost == broj)
       /* Adresu repa liste treba sacuvati pre oslobadjanja cvora na
          adresi adresa_glave. */
       pomocni = (*adresa_glave)->sledeci;
       free(*adresa_glave);
       *adresa_glave = pomocni;
     /* Ako je nakon ovog brisanja lista ostala prazna, funkcija se
        prekida. Isto se radi i ukoliko glava liste sadrzi vrednost
253
        koja je veca od broja, jer kako je lista sortirana rastuce nema
        potrebe broj traziti u repu liste. */
     if (*adresa_glave == NULL || (*adresa_glave)->vrednost > broj)
       return:
257
     /* Od ovog trenutka se u svakoj iteraciji pokazivac tekuci
        pokazuje na cvor cija vrednost je manja od trazenog broja, kao
        i svim cvorovima levo od njega. Cvor se brise ako je jednak,
261
        ili, ako je razlicit, prelazi se na sledeci cvor. Ovaj postupak
        se ponavlja dok se ne dodje do poslednjeg cvora ili prvog cvora
263
        cija vrednost je veca od trazenog broja. */
     tekuci = *adresa_glave;
265
     while (tekuci->sledeci != NULL
            && tekuci->sledeci->vrednost <= broj)
267
       if (tekuci->sledeci->vrednost == broj) {
         pomocni = tekuci->sledeci;
         tekuci->sledeci = tekuci->sledeci->sledeci;
        free(pomocni);
       } else {
         /* Ne treba brisati sledeceg od tekuceg jer je manji od
            trazenog i tekuci se pomera na sledeci cvor. */
         tekuci = tekuci->sledeci;
       }
     return;
```

```
void ispisi_listu(Cvor * glava)
281
     /* Funkciji se ne salje adresa promenljive koja cuva glavu liste
        jer se lista nece menjati */
283
     putchar('[');
     /* Unutar zagrada ispisuju se vrednosti u cvorovima liste od
285
        pocetka prema kraju liste. */
     for (; glava != NULL; glava = glava->sledeci) {
287
       printf("%d", glava->vrednost);
       if (glava->sledeci != NULL)
289
         printf(", ");
291
     printf("]\n");
```

#### $main_a.c$

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include "lista.h"
5 /* 1) Glavni program */
  int main()
    /* Lista je prazna na pocetku */
    Cvor *glava = NULL;
    Cvor *trazeni = NULL;
    int broj;
    /* Testiranje funkcije za dodavanja novog broja na pocetak liste */
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", &broj) > 0) {
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
17
         memorije za nov cvor. Memoriju alociranu za cvorove liste
         treba osloboditi pre napustanja programa. */
19
      if (dodaj_na_pocetak_liste(&glava, broj) == 1) {
        fprintf(stderr, "Greska: Neuspesna alokacija memorije za cvor.\
      n");
21
        oslobodi_listu(&glava);
        exit(EXIT_FAILURE);
      printf("\tLista: ");
      ispisi_listu(glava);
25
    /* Testiranje funkcije za pretragu liste */
    printf("\nUnesite broj koji se trazi: ");
29
    scanf("%d", &broj);
31
```

```
trazeni = pretrazi_listu(glava, broj);
if (trazeni == NULL)
    printf("Broj %d se ne nalazi u listi!\n", broj);
else
    printf("Trazeni broj %d je u listi!\n", trazeni->vrednost);

/* Oslobadja se memorija zauzeta listom */
oslobodi_listu(&glava);

exit(EXIT_SUCCESS);
}
```

#### $main_b.c$

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "lista.h"
  /* 2) Glavni program */
6 int main()
    /* Lista je prazna na pocetku */
    Cvor *glava = NULL;
    int broj;
12
    /* Testira se funkcija za dodavanja novog broja na kraj liste */
    printf("Unesite brojeve: (za kraj CTRL+D)\n");
    while (scanf("%d", &broj) > 0) {
14
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
         memorije za nov cvor. Memoriju alociranu za cvorove liste
16
         treba osloboditi pre napustanja programa. */
      if (dodaj_na_kraj_liste(&glava, broj) == 1) {
18
        fprintf(stderr, "Greska: Neuspesna alokacija memorije za cvor %
      d\n", broj);
        oslobodi_listu(&glava);
20
        exit(EXIT_FAILURE);
      printf("\tLista: ");
      ispisi_listu(glava);
24
26
    /* Testira se funkcije kojom se brise cvor liste */
    printf("\nUnesite broj koji se brise: ");
28
    scanf("%d", &broj);
30
    /* Brisu se cvorovi iz liste cije je polje vrednost jednako broju
       procitanom sa ulaza */
    obrisi_cvor(&glava, broj);
34
    printf("Lista nakon brisanja: ");
    ispisi_listu(glava);
36
```

```
/* Oslobadja se memorija zauzeta listom */
oslobodi_listu(&glava);

exit(EXIT_SUCCESS);
}
```

 $main_c.c$ 

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "lista.h"
  /* 3) Glavni program */
6 int main()
    /* Lista je prazna na pocetku */
    Cvor *glava = NULL;
    Cvor *trazeni = NULL;
    int broj;
12
    /* Testira se funkcija za dodavanje vrednosti u listu tako da bude
       uredjena neopadajuce */
14
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", \&broj) > 0) {
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
         memorije za nov cvor. Memoriju alociranu za cvorove liste
18
         treba osloboditi pre napustanja programa. */
      if (dodaj_sortirano(&glava, broj) == 1) {
        fprintf(stderr, "Greska: Neuspesna alokacija memorije za cvor %
      d\n", broj);
        oslobodi_listu(&glava);
        exit(EXIT_FAILURE);
      }
24
      printf("\tLista: ");
26
      ispisi_listu(glava);
    /* Testira se funkcija za pretragu liste */
    printf("\nUnesite broj koji se trazi: ");
    scanf("%d", &broj);
32
    trazeni = pretrazi_listu(glava, broj);
    if (trazeni == NULL)
34
      printf("Broj %d se ne nalazi u listi!\n", broj);
36
      printf("Trazeni broj %d je u listi!\n", trazeni->vrednost);
38
    /* Testira se funkcija kojom se brise cvor liste */
    printf("\nUnesite broj koji se brise: ");
40
    scanf("%d", &broj);
```

```
/* Brisu se cvorovi iz liste cije polje vrednost je jednako broju procitanom sa ulaza */
obrisi_cvor_sortirane_liste(&glava, broj);

printf("Lista nakon brisanja: ");
ispisi_listu(glava);

/* Oslobadja se memorija zauzeta listom */
oslobodi_listu(&glava);

exit(EXIT_SUCCESS);
}
```

## lista.h

```
#ifndef _LISTA_H_
2 #define _LISTA_H_
4 /* Struktura kojom je predstavljen cvor liste sadrzi celobrojni
    podatak vrednost i pokazivac na sledeci cvor liste. */
 typedef struct cvor {
   int vrednost;
   struct cvor *sledeci;
  } Cvor;
  /* Funkcija kreira cvor, vrednost novog cvora inicijalizuje na broj,
    dok pokazivac na sledeci cvor postavlja na NULL. Vraca pokazivac
     na novokreirani cvor ili NULL ako alokacija nije bila uspesna. */
14 Cvor *napravi_cvor(int broj);
16 /* Funkcija oslobadja dinamicku memoriju zauzetu za cvorove liste
     ciji se pokazivac glava nalazi na adresi adresa_glave. */
void oslobodi_listu(Cvor ** adresa_glave);
20 /* Funkcija dodaje novi cvor na pocetak liste. Vraca 1 ukoliko je
     bilo greske pri alokaciji memorije, inace vraca 0. */
22 int dodaj_na_pocetak_liste(Cvor ** adresa_glave, int broj);
24 /* Funkcija dodaje broj na kraj liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije, inace vraca 0. */
26 int dodaj_na_kraj_liste(Cvor ** adresa_glave, int broj);
28 /* Funkcija dodaje broj u rastuce sortiranu listu tako da nova lista
     ostane sortirana. Vraca 1 ukoliko je bilo greske pri alokaciji
    memorije, inace vraca 0. */
  int dodaj_sortirano(Cvor ** adresa_glave, int broj);
32
```

```
/* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
     Vraca pokazivac na cvor liste u kome je sadrzan trazeni broj ili
     NULL u slucaju da takav cvor ne postoji u listi. */
36 Cvor *pretrazi_listu(Cvor * glava, int broj);
  /* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
     U pretrazi oslanja se na cinjenicu da je lista koja se pretrazuje
     neopadajuce sortirana. Vraca pokazivac na cvor liste u kome je
     sadrzan trazeni broj ili NULL u slucaju da takav cvor ne postoji.
      */
42 Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj);
  /* Funkcija brise iz liste sve cvorove koji sadrze dati broj.
     Azurira pokazivac na glavu liste, koji moze biti promenjen u
     slucaju da se obrise stara glava liste. */
  void obrisi_cvor(Cvor ** adresa_glave, int broj);
48
  /* Funkcija brise iz liste sve cvorove koji sadrze dati broj,
     oslanjajuci se na cinjenicu da je prosledjena lista sortirana
     neopadajuce. Azurira pokazivac na glavu liste, koji moze biti
     promenjen ukoliko se obrise stara glava liste. */
  void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, int broj);
54
  /* Funkcija ispisuje samo vrednosti cvorova liste razdvojene
     zapetama. */
56
  void ispisi_vrednosti(Cvor * glava);
58
  /* Funkcija prikazuje vrednosti cvorova liste pocev od glave ka
     kraju liste, razdvojene zapetama i uokvirene zagradama. */
60
  void ispisi_listu(Cvor * glava);
62
  #endif
```

#### lista.c

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

Cvor *napravi_cvor(int broj)
{
    /* Alokacija memorije za novi cvor uz proveru uspesnosti */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
        return NULL;

/* Inicijalizacija polja strukture */
    novi->vrednost = broj;
    novi->sledeci = NULL;

/* Vraca se adresa novog cvora */
```

```
return novi;
19
  void oslobodi_listu(Cvor ** adresa_glave)
21 | {
    /* Ako je lista vec prazna */
   if (*adresa_glave == NULL)
23
     return:
    /* Ako lista nije prazna, treba osloboditi memoriju. Treba
       osloboditi rep liste, pre oslobadjanja memorije za glavu liste
    oslobodi_listu(&(*adresa_glave)->sledeci);
    /* Nakon oslobodjenog repa, oslobadja se glava liste i azurira se
       glava u pozivajucoj funkciji tako da odgovara praznoj listi */
    free(*adresa_glave);
    *adresa_glave = NULL;
33
int dodaj_na_pocetak_liste(Cvor ** adresa_glave, int broj)
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
     return 1;
41
    /* Novi cvor se uvezuje na pocetak i postaje nova glave liste */
   novi->sledeci = *adresa_glave;
43
    *adresa_glave = novi;
45
    /* Vraca se indikator uspesnog dodavanja */
    return 0;
47
49
  int dodaj_na_kraj_liste(Cvor ** adresa_glave, int broj)
51 {
    /* Ako je lista prazna */
    if (*adresa_glave == NULL) {
53
      /* Novi cvor postaje glava liste */
      Cvor *novi = napravi_cvor(broj);
      /* Ukoliko je bilo greske pri kreiranju novog cvora, vraca se 1
      if (novi == NULL)
       return 1;
59
      /* Azuriranjem vrednosti na koju pokazuje adresa_glave, ujedno
         se azurira i pokazivacka promenljiva u pozivajucoj funkciji */
      *adresa_glave = novi;
      /* Vraca se indikator uspesnog dodavanja */
      return 0;
```

```
67
     /* Ako lista nije prazna, broj se dodaje u rep liste. */
     /* Prilikom dodavanja u listu na kraj u velikoj vecini slucajeva
        novi broj se dodaje u rep liste u rekurzivnom pozivu.
        Informaciju o uspesnosti alokacije u rekurzivnom pozivu
        funkcija prosledjuje visem rekurzivnom pozivu koji tu
        informaciju vraca u rekurzivni poziv iznad, sve dok se ne vrati
        u main. Tek je iz main funkcije moguce pristupiti pravom
        pocetku liste i osloboditi je celu, ako ima potrebe. Ako je
        funkcija vratila 0, onda nije bilo greske. */
     return dodaj_na_kraj_liste(&(*adresa_glave)->sledeci, broj);
   int dodaj_sortirano(Cvor ** adresa_glave, int broj)
81
     /* Ako je lista prazna */
83
     if (*adresa_glave == NULL) {
85
       /* Novi cvor postaje glava liste */
       Cvor *novi = napravi_cvor(broj);
       /* Ukoliko je bilo greske pri kreiranju novog cvora, vraca se 1
89
       if (novi == NULL)
        return 1:
91
       /* Azurira se glava liste */
       *adresa_glave = novi;
95
       /* Vraca se indikator uspesnog dodavanja */
       return 0;
97
99
     /* Lista nije prazna. Ako je broj manji ili jednak od vrednosti u
        glavi liste, onda se dodaje na pocetak liste */
     if ((*adresa_glave)->vrednost >= broj)
       return dodaj_na_pocetak_liste(adresa_glave, broj);
     /* Inace, broj treba dodati u rep, tako da rep i sa novim cvorom
        bude sortirana lista. */
     return dodaj_sortirano(&(*adresa_glave)->sledeci, broj);
109
   Cvor *pretrazi_listu(Cvor * glava, int broj)
     /* U praznoj listi nema vrednosti */
     if (glava == NULL)
      return NULL;
     /* Ako glava liste sadrzi trazeni broj, vraca se pokazivac glava */
     if (glava->vrednost == broj)
```

```
return glava;
119
     /* Inace, pretraga se nastavlja u repu liste */
    return pretrazi_listu(glava->sledeci, broj);
   Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj)
125 4
     /* Trazenog broja nema ako je lista prazna ili ako je broj manji
       od vrednosti u glavi liste, jer je lista neopadajuce sortirana
     if (glava == NULL || glava->vrednost > broj)
      return NULL:
     /* Ako glava liste sadrzi trazeni broj, vraca se pokazivac glava */
     if (glava->vrednost == broj)
      return glava;
     /* Inace, pretraga se nastavlja u repu liste */
    return pretrazi_listu(glava->sledeci, broj);
137
  void obrisi_cvor(Cvor ** adresa_glave, int broj)
139
     /* U praznoj listi nema cvorova za brisanje. */
141
     if (*adresa_glave == NULL)
      return:
143
     /* Prvo se brisu cvorovi iz repa koji imaju vrednost broj */
145
     obrisi_cvor(&(*adresa_glave)->sledeci, broj);
147
     /* Preostaje provera da li glavu liste treba obrisati */
     if ((*adresa_glave)->vrednost == broj) {
       /* Pomocni pokazuje na cvor koji treba da se obrise */
       Cvor *pomocni = *adresa_glave;
       /* Azurira se pokazivac na glavu da pokazuje na sledeci u listi
          i brise se cvor koji je bio glava liste. */
       *adresa_glave = (*adresa_glave)->sledeci;
       free(pomocni);
  }
157
  void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, int broj)
159
     /* Ako je lista prazna ili glava liste sadrzi vrednost koja je
161
        veca od broja, kako je lista sortirana rastuce nema potrebe
        broj traziti u repu liste i zato se funkcija prekida */
     if (*adresa_glave == NULL || (*adresa_glave)->vrednost > broj)
      return;
     /* Brisu se cvorovi iz repa koji imaju vrednost broj */
167
     obrisi_cvor(&(*adresa_glave)->sledeci, broj);
```

```
/* Preostaje provera da li glavu liste treba obrisati */
     if ((*adresa_glave)->vrednost == broj) {
       /* Pomocni pokazuje na cvor koji treba da se obrise */
       Cvor *pomocni = *adresa_glave;
       /* Azurira se pokazivac na glavu da pokazuje na sledeci u listi i
          brise se cvor koji je bio glava liste */
       *adresa_glave = (*adresa_glave)->sledeci;
       free(pomocni);
   }
   void ispisi_vrednosti(Cvor * glava)
181
     /* Prazna lista */
183
     if (glava == NULL)
       return;
185
     /* Ispisuje se vrednost u glavi liste */
187
     printf("%d", glava->vrednost);
189
     /* Ako rep nije prazan, ispisuje se znak ',' i razmak. Rekurzivno
        se poziva ista funkcija za ispis ostalih. */
191
     if (glava->sledeci != NULL) {
       printf(", ");
193
       ispisi_vrednosti(glava->sledeci);
195
   void ispisi_listu(Cvor * glava)
199
     /* Funkciji se ne salje adresa promenljive koja cuva glavu liste,
        jer nece menjati listu, pa nema ni potrebe da azuriza pokazivac
201
        na glavu liste iz pozivajuce funkcije. Ona ispisuje samo
203
        zagrade, a rekurzivno ispisivanje vrednosti u listi prepusta
        rekurzivnoj pomocnoj funkciji ispisi_vrednosti, koja ce ispisati
205
        elemente razdvojene zapetom i razmakom. */
     putchar('[');
     ispisi_vrednosti(glava);
     printf("]\n");
   3
209
```

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

# #define MAX_DUZINA 20

/* Struktura kojom je predstavljen cvor liste sadrzi naziv etikete,
```

```
broj pojavljivanja etikete i pokazivac na sledeci cvor liste */
  typedef struct _Cvor {
    char etiketa[20];
   unsigned broj_pojavljivanja;
   struct _Cvor *sledeci;
12
  } Cvor:
14
  /* Pomocna funkcija koja kreira cvor. Vraca pokazivac na novi cvor
    ili NULL ako alokacija nije uspesno izvrsena */
16
  Cvor *napravi_cvor(unsigned br, char *etiketa)
18 4
    /* Alokacija memorije za cvor uz proveru uspesnosti alokacije */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor)):
20
    if (novi == NULL)
     return NULL;
    /* Inicijalizacija polja strukture */
    novi->broj_pojavljivanja = br;
    strcpy(novi->etiketa, etiketa);
26
    novi->sledeci = NULL;
28
    /* Vraca se adresa novog cvora */
    return novi;
30
  /* Funkcija oslobadja dinamicku memoriju zauzetu cvorovima liste */
void oslobodi_listu(Cvor ** adresa_glave)
    Cvor *pomocni = NULL;
36
    /* Sve dok lista ni bude prazna, brise se tekuca glava liste i
38
       azurira se vrednost glave liste */
    while (*adresa_glave != NULL) {
40
      pomocni = (*adresa_glave)->sledeci;
42
      free(*adresa_glave);
      *adresa_glave = pomocni;
44
    /* Pokazivac glava u main funkciji, na adresi adresa_glave, bice
       postavljen na NULL vrednost po izlasku iz petlje. */
46
  }
48
  /* Funkcija dodaje novi cvor na pocetak liste. Vraca 1 ako je doslo
     do greske pri alokaciji memorije za nov cvor, inace vraca 0. */
  int dodaj_na_pocetak_liste(Cvor ** adresa_glave, unsigned br,
                              char *etiketa)
52
    /* Kreira se novi cvor i proverava se uspesnost alokacije */
54
    Cvor *novi = napravi_cvor(br, etiketa);
    if (novi == NULL)
      return 1:
58
    /* Dodaje se novi cvor na pocetak liste */
```

```
novi->sledeci = *adresa_glave;
60
     *adresa_glave = novi;
62
     /* Vraca se indikator uspesnog dodavanja */
     return 0:
64
   3
66
   /* Funkcija vraca cvor koji kao vrednost sadrzi trazenu etiketu ili
     NULL ako takav cvor ne postoji u listi. */
68
   Cvor *pretrazi_listu(Cvor * glava, char etiketa[])
   {
     Cvor *tekuci;
     /* Obilazi se cvor po cvor liste */
     for (tekuci = glava; tekuci != NULL; tekuci = tekuci->sledeci)
74
       /* Ako tekuci cvor sadrzi trazenu etiketu, vraca se njegova
          vrednost */
       if (strcmp(tekuci->etiketa, etiketa) == 0)
         return tekuci;
78
     /* Cvor nije pronadjen */
80
     return NULL;
   }
82
   /* Funkcija ispisuje sadrzaj liste */
84
   void ispisi_listu(Cvor * glava)
86
     /* Pocevsi od cvora koji je glava lista, ispisuju se sve etikete i
        broj njihovog pojavljivanja u HTML datoteci. */
88
     for (; glava != NULL; glava = glava->sledeci)
       printf("%s - %u\n", glava->etiketa, glava->broj_pojavljivanja);
90
92
   /* Glavni program */
   int main(int argc, char **argv)
94
96
     /* Proverava se da li je program pozvan sa ispravnim brojem
       argumenata
        komandne linije. */
     if (argc != 2) {
       fprintf(stderr,
               "Greska: Program se poziva sa: ./a.out datoteka.html!\n")
       exit(EXIT_FAILURE);
     /* Priprema datoteke za citanje */
104
     FILE *in = NULL;
     in = fopen(argv[1], "r");
106
     if (in == NULL) {
       fprintf(stderr,
108
               "Greska: Neuspesno otvaranje datoteke %s!\n", argv[1]);
```

```
110
       exit(EXIT_FAILURE);
     char c;
     int i = 0:
114
     char procitana[MAX_DUZINA];
     Cvor *glava = NULL;
     Cvor *trazeni = NULL;
118
     /* Cita se datoteka, karakter po karakter, dok se ne procita cela
       */
     while ((c = fgetc(in)) != EOF) {
120
       /* Proverava se da li se pocinje sa citanjem nove etikete */
       if (c == '<') {
         /* Proverava se da li se cita zatvarajuca etiketa */
         if ((c = fgetc(in)) == '/') {
124
           i = 0;
           while ((c = fgetc(in)) != '>')
126
             procitana[i++] = c;
128
         /* Cita se otvarajuca etiketa */
         else {
130
           i = 0;
           procitana[i++] = c;
           while ((c = fgetc(in)) != ' ' && c != '>')
             procitana[i++] = c;
134
         procitana[i] = '\0';
136
         /* Trazi se procitana etiketa medju postojecim cvorovima
138
            liste. Ukoliko ne postoji, dodaje se novi cvor za ucitanu
            etiketu sa brojem pojavljivanja 1. Inace se uvecava broj
140
            pojavljivanja etikete. */
         trazeni = pretrazi_listu(glava, procitana);
142
         if (trazeni == NULL) {
           if (dodaj_na_pocetak_liste(&glava, 1, procitana) == 1) {
             fprintf(stderr,
                      "Greska: Neuspesna alokacija memorije za nov cvor\n
146
       ");
             oslobodi_listu(&glava);
             exit(EXIT_FAILURE);
           }
         } else
150
           trazeni->broj_pojavljivanja++;
       }
     }
154
     /* Zatvara se datoteka */
     fclose(in);
156
     /* Ispisuje se sadrzaj cvorova liste */
158
     ispisi_listu(glava);
```

```
/* Oslobadja se memorija zauzeta listom */
oslobodi_listu(&glava);

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #define MAX_INDEKS 11
6 #define MAX_IME_PREZIME 21
  /* Struktura kojom se predstavlja cvor liste koji sadrzi podatke o
     studentu */
10 typedef struct _Cvor {
    char broj_indeksa[MAX_INDEKS];
    char ime[MAX_IME_PREZIME];
    char prezime[MAX_IME_PREZIME];
    struct _Cvor *sledeci;
  } Cvor;
  /* Funkcija kreira i inicijalizuje cvor liste i vraca pokazivac na
    novi cvor ili NULL ukoliko je doslo do greske */
  Cvor *napravi_cvor(char *broj_indeksa, char *ime, char *prezime)
20 | {
    /* Alokacija memorije za cvor uz proveru uspesnosti alokacije */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
      return NULL;
    /* Inicijalizacija polja strukture */
    strcpy(novi->broj_indeksa, broj_indeksa);
    strcpy(novi->ime, ime);
    strcpy(novi->prezime, prezime);
    novi->sledeci = NULL;
30
    /* Vraca se adresa novog cvora */
    return novi;
34 }
  /* Funkcija oslobadja memoriju zauzetu cvorovima liste */
  void oslobodi_listu(Cvor ** adresa_glave)
38 {
    /* Ako je lista prazna, nema potrebe oslobadjati memoriju */
    if (*adresa_glave == NULL)
40
      return;
42
```

```
/* Rekurzivnim pozivom se oslobadja rep liste */
    oslobodi_listu(&(*adresa_glave)->sledeci);
44
    /* Potom se oslobadja i glava liste */
46
    free(*adresa_glave);
48
    /* Proglasava se lista praznom */
    *adresa_glave = NULL;
  /* Funkcija dodaje novi cvor na pocetak liste. Vraca 1 ako je doslo
    do greske pri alokaciji memorije za nov cvor, inace vraca 0. */
  int dodaj_na_pocetak_liste(Cvor ** adresa_glave, char *broj_indeksa,
                             char *ime, char *prezime)
56
    /* Kreira se novi cvor i proverava se uspesnost alokacije */
58
    Cvor *novi = napravi_cvor(broj_indeksa, ime, prezime);
    if (novi == NULL)
     return 1;
    /* Dodaje se novi cvor na pocetak liste */
   novi->sledeci = *adresa_glave;
64
    *adresa_glave = novi;
    /* Vraca se indikator uspesnog dodavanja */
    return 0;
68
  /* Funkcija ispisuje sadrzaj cvorova liste. */
72 void ispisi_listu(Cvor * glava)
    /* Pocevsi od glave liste */
74
   for (; glava != NULL; glava = glava->sledeci)
     printf("%s %s %s\n", glava->broj_indeksa, glava->ime,
             glava->prezime);
78 }
80 /* Funkcija vraca cvor koji kao vrednost sadrzi trazeni broj
     indeksa. U suprotnom funkcija vraca NULL */
82 Cvor *pretrazi_listu(Cvor * glava, char *broj_indeksa)
    /* Ako je lista prazna, ne postoji trazeni cvor */
84
    if (glava == NULL)
     return NULL;
86
    /* Poredi se trazeni broj indeksa sa brojem indeksa u glavi liste
88
    if (!strcmp(glava->broj_indeksa, broj_indeksa))
90
     return glava;
    /* Ukoliko u glavi liste nije trazeni indeks, pretraga se
92
       nastavlja u repu liste */
```

```
94
     return pretrazi_listu(glava->sledeci, broj_indeksa);
96
   /* Glavni program */
   int main(int argc, char **argv)
98
     /* Argumenti komandne linije su neophodni jer se iz komandne
100
        linije dobija ime datoteke sa informacijama o studentima */
     if (argc != 2) {
       fprintf(stderr,
               "Greska: Program se poziva sa: ./a.out ime_datoteke\n");
104
       exit(EXIT_FAILURE);
106
     /* Priprema datoteke za citanje */
108
     FILE *in = NULL:
     in = fopen(argv[1], "r");
     if (in == NULL) {
       fprintf(stderr,
                "Greska: Neuspesno otvaranje datoteke %s.\n", argv[1]);
       exit(EXIT_FAILURE);
114
     /* Deklaracije pomocnih promenljiva za citanje vrednosti koje
        treba smestiti u listu */
118
     char ime[MAX_IME_PREZIME], prezime[MAX_IME_PREZIME];
     char broj_indeksa[MAX_INDEKS];
120
     Cvor *glava = NULL;
     Cvor *trazeni = NULL;
     /* Ucitavanje vrednosti u listu */
124
     while (fscanf(in, "%s %s %s", broj_indeksa, ime, prezime) != EOF)
       if (dodaj_na_pocetak_liste(&glava, broj_indeksa, ime, prezime)) {
126
         fprintf(stderr,
128
                  "Greska: Neuspesna alokacija memorije za nov cvor\n");
         oslobodi_listu(&glava);
         exit(EXIT_FAILURE);
130
     /* Datoteka vise nije potrebna i zatvara se. */
     fclose(in);
134
     /* Ucitava se indeks po indeks studenta koji se trazi u listi. */
     while (scanf("%s", broj_indeksa) != EOF) {
       trazeni = pretrazi_listu(glava, broj_indeksa);
138
       if (trazeni == NULL)
         printf("ne\n");
140
       else
         printf("da: %s %s\n", trazeni->ime, trazeni->prezime);
144
     /* Oslobadja se memorija zauzeta za cvorove liste. */
```

```
oslobodi_listu(&glava);

exit(EXIT_SUCCESS);
}
```

Napomena: Rešenje koristi biblioteku za rad sa listama iz zadatka 4.1.

```
#include <stdio.h>
  #include <stdlib.h>
 #include "lista.h"
  /* Funkcija objedinjuje dve liste ciji se pokazivaci na glave nalaze
     na adresama adresa_glave_1 i adresa_glave_2 prevezivanjem
     pokazivaca postojecih cvorova listi */
  Cvor *objedini(Cvor ** adresa_glave_1, Cvor ** adresa_glave_2)
    /* Pokazivaci na pocetne cvorove listi koje se prevezuju */
    Cvor *lista1 = *adresa_glave_1;
    Cvor *lista2 = *adresa_glave_2;
13
    /* Pokazivac na pocetni cvor rezultujuce liste */
    Cvor *rezultujuca = NULL;
    Cvor *tekuci = NULL;
17
    /* Ako su obe liste prazne i rezultat je prazna lista */
    if (lista1 == NULL && lista2 == NULL)
19
      return NULL;
    /* Ako je prva lista prazna, rezultat je druga lista */
    if (lista1 == NULL)
      return lista2;
    /* Ako je druga lista prazna, rezultat je prva lista */
27
    if (lista2 == NULL)
      return lista1;
29
    /* Odredjuje se prvi cvor rezultujuce liste - to je ili prvi cvor
       liste lista1 ili prvi cvor liste lista2 u zavisnosti od toga
       koji sadrzi manju vrednost */
    if (lista1->vrednost < lista2->vrednost) {
      rezultujuca = lista1;
      lista1 = lista1->sledeci;
35
    } else {
      rezultujuca = lista2;
37
      lista2 = lista2->sledeci;
    }
39
    /* Kako promenljiva rezultujuca pokazuje na pocetak nove liste, ne
       sme joj se menjati vrednost. Zato se koristi pokazivac tekuci
41
       koji sadrzi adresu trenutnog cvora rezultujuce liste */
```

```
tekuci = rezultujuca;
43
    /* U svakoj iteraciji petlje rezultujucoj listi se dodaje novi
45
       cvor tako da bude uredjena neopadajuce. Pokazivac na listu iz
       koje se uzima cvor se azurira tako da pokazuje na sledeci cvor
47
      */
    while (lista1 != NULL && lista2 != NULL) {
      if (lista1->vrednost < lista2->vrednost) {
49
        tekuci->sledeci = lista1;
        lista1 = lista1->sledeci;
      } else {
        tekuci->sledeci = lista2;
53
        lista2 = lista2->sledeci;
      tekuci = tekuci->sledeci;
    /* Ako se iz petlje izaslo jer se stiglo do kraja prve liste, na
       rezultujucu listu treba nadovezati ostatak druge liste */
    if (lista1 == NULL)
      tekuci->sledeci = lista2;
    else
      /* U suprotnom treba nadovezati ostatak prve liste */
      tekuci->sledeci = lista1;
    /* Preko adresa glava polaznih listi vrednosti pokazivaca u
       pozivajucoj funkciji se postavljaju na NULL jer se svi cvorovi
       prethodnih listi nalaze negde unutar rezultujuce liste. Do njih
       se moze doci prateci pokazivace iz glave rezultujuce liste, tako
       da stare pokazivace treba postaviti na NULL. */
    *adresa_glave_1 = NULL;
    *adresa_glave_2 = NULL;
73
    return rezultujuca;
  int main(int argc, char **argv)
79
    /* Argumenti komandne linije su neophodni */
    if (argc != 3) {
81
      fprintf(stderr,
               "Greska: Program se poziva sa:\n ./a.out dat1.txt dat2.
83
      txt\n");
      exit(EXIT_FAILURE);
85
    /* Otvaraju se datoteke u kojima se nalaze elementi listi */
    FILE *in1 = NULL;
    in1 = fopen(argv[1], "r");
89
    if (in1 == NULL) {
      fprintf(stderr,
91
               "Greska: Neuspesno otvaranje datoteke %s.\n", argv[1]);
```

```
93
       exit(EXIT_FAILURE);
95
     FILE *in2 = NULL;
     in2 = fopen(argv[2], "r");
97
     if (in2 == NULL) {
       fprintf(stderr,
99
               "Greska: Neuspesno otvaranje datoteke %s.\n", argv[2]);
       exit(EXIT_FAILURE);
     /* Liste su na pocetku prazne */
     int broj;
     Cvor *lista1 = NULL;
     Cvor *lista2 = NULL;
     /* Ucitavanje listi */
     while (fscanf(in1, "%d", &broj) != EOF)
       dodaj_na_kraj_liste(&lista1, broj);
    while (fscanf(in2, "%d", &broj) != EOF)
113
       dodaj_na_kraj_liste(&lista2, broj);
     /* Datoteke vise nisu potrebne i treba ih zatvoriti. */
     fclose(in1);
117
     fclose(in2);
119
     /* Pokazivac rezultat ce pokazivati na glavu liste dobijene
       objedinjavanjem listi */
     Cvor *rezultat = objedini(&lista1, &lista2);
123
     /* Ispis rezultujuce liste. */
     ispisi_listu(rezultat);
127
     /* Lista rezultat dobijena je prevezivanjem cvorova polaznih listi.
        Njenim oslobadjanjem oslobadja se sva zauzeta memorija. */
     oslobodi_listu(&rezultat);
129
     exit(EXIT_SUCCESS);
131
```

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

/* Struktura kojom je predstavljen cvor liste sadrzi karakter koji
predstavlja vidjenu zagradu i pokazivac na sledeci cvor liste */
typedef struct cvor {
   char zagrada;
   struct cvor *sledeci;
```

```
9 \ Cvor;
11 /* Funkcija koja oslobadja memoriju zauzetu stekom */
  void oslobodi_stek(Cvor ** stek)
13 | {
    Cvor *tekuci:
    Cvor *pomocni;
    /* Oslobadja se cvor po cvor steka */
17
    tekuci = *stek;
    while (tekuci != NULL) {
19
      pomocni = tekuci->sledeci;
      free(tekuci);
      tekuci = pomocni;
23
    /* Stek se proglasava praznim */
    *stek = NULL;
  /* Glavni program */
  int main()
31
    /* Stek je na pocetku prazan */
    Cvor *stek = NULL;
33
    FILE *ulaz = NULL;
    char c:
35
    Cvor *pomocni = NULL;
37
    /* Otvaranje datotoke za citanje izraza */
    ulaz = fopen("izraz.txt", "r");
39
    if (ulaz == NULL) {
      fprintf(stderr,
41
               "Greska: Neuspesno otvaranje datoteke izraz.txt!\n");
      exit(EXIT_FAILURE);
43
45
    /* Cita se datoteka, karakter po karakter, dok se ne procita cela
    while ((c = fgetc(ulaz)) != EOF) {
47
      /* Ako je ucitana otvorena zagrada, stavlja se na stek */
      if (c == '(' || c == '{' || c == '[') {
49
        /* Alocira se memorija za novi cvor liste i proverava se
           uspesnost alokacije */
        pomocni = (Cvor *) malloc(sizeof(Cvor));
        if (pomocni == NULL) {
          fprintf(stderr, "Greska: Neuspesna alokacija memorije!\n");
           /* Oslobadjanje memorije zauzete stekom */
          oslobodi_stek(&stek);
          /* Prekid izvrsavanja programa */
          exit(EXIT_FAILURE);
59
```

```
/* Inicijalizacija polja strukture */
61
         pomocni->zagrada = c;
         /* Promena vrha steka */
         pomocni->sledeci = stek;
         stek = pomocni;
       /* Ako je ucitana zatvorena zagrada, proverava se da li je stek
          prazan i ako nije, da li se na vrhu steka nalazi odgovarajuca
          otvorena zagrada */
       else {
         if (c == ')' || c == '}' || c == ']') {
           if (stek != NULL && ((stek->zagrada == '(' && c == ')')
73
                                 || (stek->zagrada == '{' && c == '}')
                                 || (stek->zagrada == '[' && c == ']')))
       {
             /* Sa vrha steka se uklanja otvorena zagrada */
             pomocni = stek->sledeci;
             free(stek);
             stek = pomocni;
           } else {
             /* Dakle zagrade u izrazu nisu ispravno uparene */
             break:
           }
83
         }
       }
85
87
     /* Procitana je cela datoteka i treba je zatvoriti */
     fclose(ulaz);
89
     /* Ako je stek prazan i procitana je cela datoteka, zagrade su
91
        ispravno uparene */
     if (stek == NULL && c == EOF)
       printf("Zagrade su ispravno uparene.\n");
     else {
95
      /* U suprotnom se zakljucuje da zagrade nisu ispravno uparene */
       printf("Zagrade nisu ispravno uparene.\n");
       /* Oslobadja se memorija koja je ostala zauzeta stekom */
       oslobodi_stek(&stek);
99
     exit(EXIT_SUCCESS);
103 }
```

## stek.h

```
#ifndef _STEK_H_
2 #define _STEK_H_
4 #include <stdio.h>
  #include <stdlib.h>
6 #include <string.h>
  #include <ctype.h>
  #define MAX 100
  #define OTVORENA 1
12 #define ZATVORENA 2
14 #define VAN_ETIKETE 0
  #define PROCITANO_MANJE 1
16 #define U_ETIKETI 2
18 /* Struktura kojim se predstavlja cvor liste sadrzi ime etikete i
     pokazivac na sledeci cvor */
20 typedef struct cvor {
    char etiketa[MAX];
    struct cvor *sledeci;
  } Cvor;
  /* Funkcija kreira novi cvor, upisuje u njega etiketu i vraca
    njegovu adresu ili NULL ako alokacija nije bila uspesna */
  Cvor *napravi_cvor(char *etiketa);
  /* Funkcija oslobadja memoriju zauzetu stekom */
30 void oslobodi_stek(Cvor ** adresa_vrha);
32 /* Funkcija postavlja na vrh steka novu etiketu. U slucaju greske
     pri alokaciji za novi cvor funkcija vraca 1, inace vraca 0 */
34 int potisni_na_stek(Cvor ** adresa_vrha, char *etiketa);
  /* Funkcija skida sa vrha steka etiketu. Ako je drugi argument
     pokazivac razlicit od NULL, tada u niz karaktera na koji on
     pokazuje upisuje ime etikete koja je upravo skinuta sa steka dok
38
     u suprotnom ne radi nista. Funkcija vraca 0 ako je stek prazan
     (pa samim tim nije bilo moguce skinuti vrednost sa steka) ili 1 u
     suprotnom */
42 int skini_sa_steka(Cvor ** adresa_vrha, char *etiketa);
44 /* Funkcija vraca pokazivac na string koji sadrzi etiketu na vrhu
     steka. Ukoliko je stek prazan, vraca NULL */
46 char *vrh steka(Cvor * vrh);
```

```
/* Funkcija prikazuje stek od vrha prema dnu */
void prikazi_stek(Cvor * vrh);

/* Funkcija iz datoteke kojoj odgovara pokazivac f cita sledecu
etiketu, i upisuje je u nisku na koju pokazuje pokazivac etiketa.
Vraca EOF u slucaju da se dodje do kraja datoteke pre nego sto se
procita etiketa. Vraca OTVORENA, ako je procitana otvorena
etiketa, odnosno ZATVORENA, ako je procitana zatvorena etiketa */
int uzmi_etiketu(FILE * f, char *etiketa);

#endif
```

stek.c

```
#include "stek.h"
  Cvor *napravi_cvor(char *etiketa)
    /* Alokacija memorije za novi cvor uz proveru uspesnost alokacije
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
6
    if (novi == NULL)
      return NULL;
    /* Inicijalizacija polja u novom cvoru */
    if (strlen(etiketa) >= MAX) {
      fprintf(stderr, "Greska: Etiketa je preduga, bice skracena.\n");
12
      etiketa[MAX - 1] = ' \setminus 0';
    strcpy(novi->etiketa, etiketa);
16
    novi->sledeci = NULL;
    /* Vraca se adresa novog cvora */
18
    return novi;
20 }
void oslobodi_stek(Cvor ** adresa_vrha)
  {
    Cvor *pomocni;
24
26
    /* Sve dok stek nije prazan, brise se cvor koji je vrh steka */
    while (*adresa_vrha != NULL) {
      /* Potrebno je prvo zapamtiti adresu sledeceg cvora i onda
28
         osloboditi cvor koji predstavlja vrh steka */
      pomocni = *adresa_vrha;
30
      /* Sledeci cvor je novi vrh steka */
      *adresa_vrha = (*adresa_vrha)->sledeci;
      free(pomocni);
34
    /* Nakon izlaska iz petlje stek je prazan i pokazivac na adresi
36
```

```
adresa_vrha ce pokazivati na NULL. */
  }
38
40 int potisni_na_stek(Cvor ** adresa_vrha, char *etiketa)
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
42
    Cvor *novi = napravi_cvor(etiketa);
    if (novi == NULL)
44
      return 1;
46
    /* Novi cvor se uvezuje na vrh i postaje nov vrh steka */
    novi->sledeci = *adresa_vrha;
48
    *adresa_vrha = novi;
    return 0;
52
  int skini_sa_steka(Cvor ** adresa_vrha, char *etiketa)
54
    Cvor *pomocni;
56
    /* Pokusaj skidanja vrednosti sa praznog steka rezultuje greskom i
       vraca se 0 */
58
    if (*adresa_vrha == NULL)
      return 0;
60
    /* Ako adresa na koju se smesta etiketa nije NULL, onda se na tu
62
       adresu kopira etiketa sa vrha steka */
    if (etiketa != NULL)
64
      strcpy(etiketa, (*adresa_vrha)->etiketa);
66
    /* Element sa vrha steka se uklanja */
    pomocni = *adresa_vrha;
68
    *adresa_vrha = (*adresa_vrha)->sledeci;
    free(pomocni);
    /* Vraca se indikator uspesno izvrsene radnje */
    return 1;
  }
  char *vrh_steka(Cvor * vrh)
    /* Prazan stek nema cvor koji je vrh i vraca se NULL */
    if (vrh == NULL)
      return NULL;
80
    /* Inace, vraca se pokazivac na nisku etiketa koja je polje cvora
       koji je na vrhu steka. */
    return vrh->etiketa;
86
  void prikazi_stek(Cvor * vrh)
```

```
/* Ispisuje se spisak etiketa na steku od vrha ka dnu. */
     for (; vrh != NULL; vrh = vrh->sledeci)
       printf("<%s>\n", vrh->etiketa);
92 }
94 int uzmi_etiketu(FILE * f, char *etiketa)
     int c:
96
     int i = 0;
     /* Stanje predstavlja informaciju dokle se stalo sa citanjem
        etikete. Inicijalizuje se vrednoscu VAN_ETIKETE jer jos uvek
        nije zapoceto citanje. */
100
     /* Tip predstavlja informaciju o tipu etikete. Uzima vrednosti
        OTVORENA ili ZATVORENA. */
     int stanje = VAN_ETIKETE;
     int tip;
104
     /* HTML je neosetljiv na razliku izmedju malih i velikih slova,
106
        dok to u C-u ne vazi. Zato ce sve etikete biti prevedene u
        zapis samo malim slovima. */
108
     while ((c = fgetc(f)) != EOF) {
       switch (stanje) {
       case VAN_ETIKETE:
         if (c == '<')
112
           stanje = PROCITANO_MANJE;
         break:
114
       case PROCITANO_MANJE:
         if (c == '/') {
           /* Cita se zatvorena etiketa */
           tip = ZATVORENA;
118
         } else {
           if (isalpha(c)) {
120
             /* Cita se otvorena etiketa */
             tip = OTVORENA;
             etiketa[i++] = tolower(c);
           }
         }
         /* Od sada se cita etiketa i zato se menja stanje */
126
         stanje = U_ETIKETI;
         break;
128
       case U_ETIKETI:
         if (isalpha(c) && i < MAX - 1) {
130
           /* Ako je procitani karakter slovo i nije prekoracena
              dozvoljena duzina etikete, procitani karakter se smanjuje
              i smesta u etiketu */
           etiketa[i++] = tolower(c);
         } else {
           /* Inace, staje se sa citanjem etikete. Korektno se zavrsava
              niska koja sadrzi procitanu etiketu i vraca se njen tip */
           etiketa[i] = '\0';
138
           return tip;
140
```

```
break;
}

/* Doslo se do kraja datoteke pre nego sto je procitana naredna
etiketa i vraca se EOF. */
return EOF;
}
```

main.c

```
1 #include "stek.h"
  /* Glavni program */
  int main(int argc, char **argv)
    /* Na pocetku, stek je prazan i etikete su uparene jer nijedna jos
       nije procitana. */
    Cvor *vrh = NULL;
    char etiketa[MAX];
    int tip;
    int uparene = 1;
    FILE *f = NULL;
13
    /* Ime datoteke se preuzima iz komandne linije. */
    if (argc < 2) {
      fprintf(stderr, "Greska:");
      fprintf(stderr, "Program se poziva sa:\n %s ime_html_datoteke\n",
17
               argv[0]);
      exit(EXIT_FAILURE);
19
    }
    /* Datoteka se otvara za citanje */
    if ((f = fopen(argv[1], "r")) == NULL) {
      fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s.\n",
              argv[1]);
      exit(EXIT_FAILURE);
    /* Cita se etiketa po etiketa, sve dok ih ima u datoteci. */
29
    while ((tip = uzmi_etiketu(f, etiketa)) != EOF) {
31
      /* Ako je otvorena etiketa, stavlja se na stek. Izuzetak su
         etikete <br/> <br/>hr> i <meta> koje nemaju sadrzaj, pa ih nije
         potrebno zatvoriti. U HTML-u postoje jos neke etikete koje
33
         koje nemaju sadrzaj (npr link). Zbog jednostavnosti
         pretpostavlja se da njih nema u HTML dokumentu. */
35
      if (tip == OTVORENA) {
        if (strcmp(etiketa, "br") != 0
37
            && strcmp(etiketa, "hr") != 0
            && strcmp(etiketa, "meta") != 0)
39
          if (potisni_na_stek(&vrh, etiketa) == 1) {
            fprintf(stderr,
```

```
"Greska: Neuspesna alokacija memorije za nov cvor\n
      ");
            oslobodi stek(&vrh):
43
            exit(EXIT_FAILURE);
45
      /* Ako je zatvorena etiketa, tada je uslov dobre uparenosti da
47
         je u pitanju zatvaranje etikete koja je poslednja otvorena, a
         jos uvek nije zatvorena. Ona se mora nalaziti na vrhu steka.
         Ako je taj uslov ispunjen, skida se sa steka, jer je upravo
         zatvorena. U suprotnom, pronadjena je nepravilnost i etikete
         nisu pravilno uparene. */
      else if (tip == ZATVORENA) {
        if (vrh_steka(vrh) != NULL
            && strcmp(vrh_steka(vrh), etiketa) == 0)
          skini_sa_steka(&vrh, NULL);
        else {
          printf("Etikete nisu pravilno uparene\n");
          printf("(nadjena je etiketa </%s>", etiketa);
          if (vrh_steka(vrh) != NULL)
            printf(", a poslednja otvorena je <%s>)\n",
                   vrh_steka(vrh));
            printf(" koja nije otvorena)\n");
          uparene = 0;
          break;
        }
      }
    }
    /* Zavrseno je citanje i datoteka se zatvara */
    fclose(f);
    /* Ako do sada nije pronadjeno pogresno uparivanje, stek bi
73
       trebalo da bude prazan. Ukoliko nije, tada postoje etikete koje
       su ostale otvorene */
    if (uparene) {
      if (vrh_steka(vrh) == NULL)
        printf("Etikete su pravilno uparene!\n");
79
      else {
        printf("Etikete nisu pravilno uparene\n");
        printf("(etiketa <%s> nije zatvorena)\n", vrh_steka(vrh));
        /* Oslobadja se memorija zauzeta stekom */
        oslobodi_stek(&vrh);
83
      }
    }
85
    exit(EXIT_SUCCESS);
87
```

## red.h

```
1 #ifndef _RED_H_
  #define _RED_H_
  #include <stdio.h>
5 #include <stdlib.h>
  #define MAX 1000
  #define JMBG_DUZINA 14
  /* Struktura predstavlja zahtev korisnika. Obuhvata JMBG korisnika i
     opis njegovog zahteva. */
  typedef struct {
    char jmbg[JMBG_DUZINA];
    char opis[MAX];
15 } Zahtev;
17 /* Struktura kojom je predstavljen cvor liste, obuhvata zahtev
     korisnika i pokazivac na sledeci cvor liste. */
19 typedef struct cvor {
    Zahtev nalog;
    struct cvor *sledeci;
  } Cvor;
  /* Funkcija kreira novi cvor, inicijalizuje polje nalog na zahtev sa
     poslate adrese i vraca adresu novog cvora ili NULL ako je doslo do
     greske pri alokaciji. Prosledjuje joj se pokazivac na zahtev koji
     treba smestiti u novi cvor zbog smestanja manjeg podatka na
     sistemski stek. Pokazivac na strukturu Zahtev je manje velicine u
     bajtovima(B) u odnosu na strukturu Zahtev. */
  Cvor *napravi_cvor(Zahtev * zahtev);
  /* Funkcija prazni red oslobadjajuci memoriju koji je red zauzimao */
void oslobodi_red(Cvor ** pocetak, Cvor ** kraj);
  /* Funkcija dodaje na kraj reda novi zahtev. Vraca 1 ako je doslo do
     greske pri alokaciji memorije za novi cvor, inace vraca 0. */
  int dodaj_u_red(Cvor ** adresa_pocetka, Cvor ** adresa_kraja,
                  Zahtev * zahtev);
  /* Funkcija skida sa pocetka reda zahtev. Ako je poslednji argument
     pokazivac razlicit od NULL, tada se u strukturu na koju on
     pokazuje upisuje zahtev koji je upravo skinut sa reda dok u
     suprotnom ne upisuje nista. Vraca 0, ako je red bio prazan ili 1
43
     u suprotnom. */
45 int skini_sa_reda(Cvor ** adresa_pocetka, Cvor ** adresa_kraja,
                    Zahtev * zahtev);
```

```
/* Funkcija vraca pokazivac na strukturu koja sadrzi zahtev
korisnika na pocetku reda. Ukoliko je red prazan funkcija vraca
NULL. */
Zahtev *pocetak_reda(Cvor * pocetak);

/* Funkcija prikazuje sadrzaj reda. */
void prikazi_red(Cvor * pocetak);

#endif
```

#### red.c

```
#include "red.h"
  Cvor *napravi_cvor(Zahtev * zahtev)
    /* Alokacija memorije za novi cvor uz proveru uspesnost alokacije
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
      return NULL;
   /* Inicijalizacija polja strukture */
   novi->nalog = *zahtev;
   novi->sledeci = NULL;
12
    /* Vraca se adresa novog cvora */
    return novi;
16 }
void oslobodi_red(Cvor ** pocetak, Cvor ** kraj)
    Cvor *pomocni = NULL;
    /* Sve dok red nije prazan brise se cvor koji je pocetka reda */
    while (*pocetak != NULL) {
      /* Potrebno je prvo zapamtiti adresu sledeceg cvora i onda
         osloboditi cvor sa pocetka reda */
      pomocni = *pocetak;
26
      *pocetak = (*pocetak)->sledeci;
28
      free(pomocni);
    /* Nakon izlaska iz petlje red je prazan. Pokazivac na kraj reda
30
       treba postaviti na NULL. */
    *kraj = NULL;
  }
34
  int dodaj_u_red(Cvor ** adresa_pocetka, Cvor ** adresa_kraja,
                  Zahtev * zahtev)
36
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
```

```
Cvor *novi = napravi_cvor(zahtev);
    if (novi == NULL)
40
      return 1:
42
    /* U red se uvek dodaje na kraj. Zbog postojanja pokazivaca na
       kraj, to je podjednako efikasno kao dodavanje na pocetak liste
44
    if (*adresa_kraja != NULL) {
      (*adresa_kraja)->sledeci = novi;
46
      *adresa_kraja = novi;
    } else {
48
      /* Ako je red bio ranije prazan */
      *adresa_pocetka = novi;
      *adresa_kraja = novi;
    /* Vraca se indikator uspesnog dodavanja */
54
    return 0;
  }
56
int skini_sa_reda(Cvor ** adresa_pocetka, Cvor ** adresa_kraja,
                     Zahtev * zahtev)
60
    Cvor *pomocni = NULL;
62
    /* Ako je red prazan */
    if (*adresa_pocetka == NULL)
64
      return 0;
66
    /* Ako je prosledjen pokazivac zahtev, na tu adresu se prepisuje
       zahtev koji je na pocetku reda. */
68
    if (zahtev != NULL)
      *zahtev = (*adresa_pocetka)->nalog;
72
    /* Oslobadja se memorija zauzeta cvorom sa pocetka reda i
       azurira se pokazivac na adresi adresa_pocetka da pokazuje na
       sledeci cvor u redu. */
74
    pomocni = *adresa_pocetka;
    *adresa_pocetka = (*adresa_pocetka)->sledeci;
76
    free(pomocni);
78
    /* Ukoliko red nakon oslobadjanja pocetnog cvora ostane prazan,
       potrebno je azurirati i vrednost pokazivaca na adresi
80
       adresa_kraja na NULL */
    if (*adresa_pocetka == NULL)
82
      *adresa_kraja = NULL;
84
    return 1;
  }
86
88 Zahtev *pocetak_reda(Cvor * pocetak)
```

```
/* U praznom redu nema zahteva */
if (pocetak == NULL)
   return NULL;

/* Inace, vraca se pokazivac na zahtev sa pocetka reda */
   return &(pocetak->nalog);
}

void prikazi_red(Cvor * pocetak)
{
   /* Prikazuje se sadrzaj reda od pocetka prema kraju */
   for (; pocetak != NULL; pocetak = pocetak->sledeci)
        printf("%s %s\n", (pocetak->nalog).jmbg, (pocetak->nalog).opis);

printf("\n");
}
```

#### main.c

```
#include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
  #include "red.h"
  #define VREME_ZA_PAUZU 5
  /* Glavni program */
9 int main(int argc, char **argv)
    /* Red je prazan. */
    Cvor *pocetak = NULL, *kraj = NULL;
   Zahtev nov_zahtev;
    Zahtev *sledeci = NULL;
   char odgovor[3];
    int broj_usluzenih = 0;
17
    /* Sluzbenik evidentira korisnicke zahteve unosenjem njihovog JMBG
19
       broja i opisa potrebne usluge. */
    printf("Sluzbenik evidentira korisnicke zahteve:\n");
21
    while (1) {
      /* Ucitava se JMBG broj */
      printf("\nNovi zahtev [CTRL+D za kraj]\n\tJMBG: ");
      if (scanf("%s", nov_zahtev.jmbg) == EOF)
        break;
27
      /* Neophodan je poziv funkcije getchar da bi se i nov red nakon
         JMBG broja procitao i da bi fgets nakon toga procitala
29
         ispravan red sa opisom zahteva */
      getchar();
```

```
/* Ucitava se opis problema */
33
      printf("\t0pis problema: ");
      fgets(nov_zahtev.opis, MAX - 1, stdin);
      /* Ako je poslednji karakter nov red, eliminise se */
      if (nov_zahtev.opis[strlen(nov_zahtev.opis) - 1] == '\n')
        nov_zahtev.opis[strlen(nov_zahtev.opis) - 1] = '\0';
39
      /* Dodaje se zahtev u red i proverava se uspesnost dodavanja */
      if (dodaj_u_red(&pocetak, &kraj, &nov_zahtev) == 1) {
41
        fprintf(stderr,
                "Greska: Neuspesna alokacija memorije za nov cvor\n");
43
        oslobodi_red(&pocetak, &kraj);
        exit(EXIT_FAILURE);
45
      }
    }
47
    /* Otvaranje datoteke za dopisivanje izvestaja */
49
    FILE *izlaz = fopen("izvestaj.txt", "a");
    if (izlaz == NULL) {
      fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke izvestaj.txt\n");
      exit(EXIT_FAILURE);
    /* Dokle god ima korisnika u redu, treba ih usluziti */
    while (1) {
      sledeci = pocetak_reda(pocetak);
      /* Ako nema nikog vise u redu, prekida se petlja */
      if (sledeci == NULL)
        break;
      printf("\nSledeci je korisnik sa JMBG: %s\n", sledeci->jmbg);
      printf("i zahtevom: %s\n", sledeci->opis);
      skini_sa_reda(&pocetak, &kraj, &nov_zahtev);
      broj_usluzenih++;
      printf("\tDa li ga vracate na kraj reda? [Da/Ne] ");
      scanf("%s", odgovor);
      if (strcmp(odgovor, "Da") == 0)
75
        dodaj_u_red(&pocetak, &kraj, &nov_zahtev);
      else
        fprintf(izlaz, "JMBG: %s\tZahtev: %s\n", nov_zahtev.jmbg,
                nov_zahtev.opis);
79
      if (broj_usluzenih == VREME_ZA_PAUZU) {
        printf("\nDa li je kraj smene? [Da/Ne] ");
81
        scanf("%s", odgovor);
83
        if (strcmp(odgovor, "Da") == 0)
```

```
85
         break;
        else
         broj_usluzenih = 0;
87
89
    91
      Usluzivanje korisnika moze da se izvrsi i na sledeci nacin: */
    while (skini_sa_reda(&pocetak, &kraj, &nov_zahtev)) {
95
        printf("\nSledeci je korisnik sa JMBG brojem: %s\n",
             nov_zahtev.jmbg);
        printf("sa zahtevom: %s\n", nov_zahtev.opis);
        broj_usluzenih++;
90
        printf("\tDa li ga vracate na kraj reda? [Da/Ne] ");
        scanf("%s", odgovor);
        if (strcmp(odgovor, "Da") == 0)
         dodaj_u_red(&pocetak, &kraj, &nov_zahtev);
        else
         fprintf(izlaz, "JMBG: %s\tZahtev: %s\n",
                nov_zahtev.jmbg, nov_zahtev.opis);
        if (broj_usluzenih == VREME_ZA_PAUZU) {
         printf("\nDa li je kraj smene? [Da/Ne] ");
         scanf("%s", odgovor);
         if (strcmp(odgovor, "Da") == 0)
           break;
         else
           broj_usluzenih = 0;
        }
117
       **********************
119
    /* Datoteka vise nije potrebna i treba je zatvoriti. */
    fclose(izlaz);
    /* Ukoliko je sluzbenik prekinuo sa radom, mozda je bilo jos
       neusluzenih korisnika, u tom slucaju treba osloboditi memoriju
123
       koju zauzima red sa neobradjenim zahtevima korisnika. */
    oslobodi_red(&pocetak, &kraj);
    exit(EXIT_SUCCESS);
127
```

 $dvostruko\_povezana\_lista.h$ 

```
#ifndef _DVOSTRUKO_POVEZANA_LISTA_H_
```

```
2 #define _DVOSTRUKO_POVEZANA_LISTA_H_
  /* Struktura kojom je predstavljen cvor liste sadrzi celobrojnu
     vrednost i pokazivace na sledeci i prethodni cvor liste. */
6 typedef struct cvor {
    int vrednost;
    struct cvor *sledeci;
    struct cvor *prethodni;
10 } Cvor;
12 /* Funkcija kreira cvor, vrednost novog cvora inicijalizuje na broj,
     dok pokazivac na sledeci cvor postavlja na NULL. Vraca pokazivac
     na novokreirani cvor ili NULL ako alokacija nije bila uspesna. */
14
  Cvor *napravi_cvor(int broj);
16
  /* Funkcija oslobadja dinamicku memoriju zauzetu za cvorove liste
     ciji se pocetni cvor nalazi na adresi adresa_glave, a poslednji
18
     na adresi adresa_kraja. */
void oslobodi_listu(Cvor ** adresa_glave, Cvor ** adresa_kraja);
22 /* Funkcija dodaje novi cvor na pocetak liste. Vraca 1 ukoliko je
     bilo greske pri alokaciji memorije, inace vraca 0. */
24 int dodaj_na_pocetak_liste(Cvor ** adresa_glave, Cvor **
                             adresa_kraja, int broj);
26
  /* Funkcija dodaje broj na kraj liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije, inace vraca 0. */
28
  int dodaj_na_kraj_liste(Cvor ** adresa_glave, Cvor ** adresa_kraja,
                          int broj);
30
  /* Pomocna funkcija pronalazi cvor u listi iza koga treba umetnuti
     novi cvor sa vrednoscu broj. */
34 Cvor *pronadji_mesto_umetanja(Cvor * glava, int broj);
  /* Funkcija dodaje broj iza zadatog cvora. Vraca 1 ukoliko je
     dodavanje uspesno, odnosno 0 ukoliko je doslo do greske. */
int dodaj_iza(Cvor * tekuci, int broj);
  /* Funkcija dodaje broj u sortiranu listu tako da lista ostane
     sortirana. Vraca 1 ukoliko je bilo greske pri alokaciji memorije,
     inace vraca 0. */
42
  int dodaj_sortirano(Cvor ** adresa_glave, Cvor ** adresa_kraja, int
                      broj);
44
  /* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
46
     Vraca pokazivac na cvor liste u kome je sadrzan trazeni broj ili
     NULL u slucaju da takav cvor ne postoji u listi. */
48
  Cvor *pretrazi_listu(Cvor * glava, int broj);
50
  /* Funkcija trazi u listi cvor cija je vrednost jednaka datom broju.
     U pretrazi oslanja se na cinjenicu da je lista koja se pretrazuje
     neopadajuce sortirana. Vraca pokazivac na cvor liste koji sadrzi
```

```
trazeni broj ili NULL u slucaju da takav cvor ne postoji. */
  Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj);
56
  /* Funkcija brise cvor na koji pokazuje pokazivac tekuci u listi
    ciji pokazivac glava se nalazi na adresi adresa_glave. */
58
  void obrisi_tekuci(Cvor ** adresa_glave, Cvor ** adresa_kraja, Cvor *
                     tekuci);
60
62 /* Funkcija brise iz liste sve cvorove koji sadrze dati broj.
     Azurira pokazivac na glavu liste, koji moze biti promenjen u
    slucaju da se obrise stara glava. */
  void obrisi_cvor(Cvor ** adresa_glave, Cvor ** adresa_kraja, int
                   broj);
66
68 /* Funkcija brise iz liste sve cvorove koji sadrze dati broj,
     oslanjajuci se na cinjenicu da je prosledjena lista neopadajuce
     sortirana. Azurira pokazivac na glavu liste, koji moze biti
     promenjen ukoliko se obrise stara glava liste. */
void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, Cvor **
                                   adresa_kraja, int broj);
74
  /* Funkcija prikazuje vrednosti cvorova liste pocev od glave ka
    kraju liste, razdvojene zapetama i uokvirene zagradama. */
  void ispisi_listu(Cvor * glava);
78
  /* Funkcija prikazuje vrednosti cvorova liste pocevsi od kraja ka
    glavi liste, razdvojene zapetama i uokvirene zagradama. */
80
  void ispisi_listu_unazad(Cvor * kraj);
82
  #endif
```

## $dvostruko\_povezana\_lista.c$

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

Cvor *napravi_cvor(int broj)
{
    /* Alokacija memorije za novi cvor uz proveru uspesnosti */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
        return NULL;

/* Inicijalizacija polja strukture */
    novi->vrednost = broj;
    novi->sledeci = NULL;

/* Vraca se adresa novog cvora */
    return novi;
}
```

```
void oslobodi_listu(Cvor ** adresa_glave, Cvor ** adresa_kraja)
  {
21
    Cvor *pomocni = NULL;
23
    /* Ako lista nije prazna, onda treba osloboditi memoriju */
    while (*adresa_glave != NULL) {
      /* Potrebno je prvo zapamtiti adresu sledeceg cvora i onda
         osloboditi memoriju cvora koji predstavlja glavu liste */
      pomocni = (*adresa_glave)->sledeci;
      free(*adresa_glave);
      /* Sledeci cvor je nova glava liste */
      *adresa_glave = pomocni;
    /* Nakon izlaska iz petlje lista je prazna. Pokazivac na kraj
33
       liste treba postaviti na NULL */
    *adresa_kraja = NULL;
35
37
  int dodaj_na_pocetak_liste(Cvor ** adresa_glave, Cvor **
                              adresa_kraja, int broj)
39
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
41
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
43
      return 1:
45
    /* Sledbenik novog cvora je glava stare liste */
    novi->sledeci = *adresa_glave;
47
    /* Ako stara lista nije bila prazna, onda prethodni cvor glave
49
       treba da bude novi cvor. Inace, novi cvor je ujedno i pocetni i
       krajnji */
    if (*adresa_glave != NULL)
53
      (*adresa_glave)->prethodni = novi;
    else
      *adresa_kraja = novi;
    /* Novi cvor je nova glava liste */
    *adresa_glave = novi;
59
    /* Vraca se indikator uspesnog dodavanja */
    return 0;
61
63
  int dodaj_na_kraj_liste(Cvor ** adresa_glave, Cvor ** adresa_kraja,
                           int broj)
65
    /* Kreira se novi cvor i proverava se uspesnost kreiranja */
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
      return 1;
```

```
71
     /* U slucaju prazne liste, glava nove liste je upravo novi cvor i
        ujedno i cela lista. Azurira se vrednost na koju pokazuju
        adresa_glave i adresa_kraja */
     if (*adresa_glave == NULL) {
       *adresa_glave = novi;
       *adresa_kraja = novi;
     } else {
       /* Ako lista nije prazna, novi cvor se dodaje na kraj liste kao
          sledbenik poslednjeg cvora i azurira se samo pokazivac na
          kraj liste */
81
       (*adresa_kraja)->sledeci = novi;
       novi->prethodni = (*adresa_kraja);
83
       *adresa_kraja = novi;
85
     /* Vraca se indikator uspesnog dodavanja */
87
    return 0;
  }
89
91 | Cvor *pronadji_mesto_umetanja(Cvor * glava, int broj)
     /* U praznoj listi nema takvog mesta i vraca se NULL */
93
     if (glava == NULL)
      return NULL;
95
     /* Pokazivac glava se pomera na sledeci cvor sve dok ne bude
97
        pokazivala na cvor ciji sledeci cvor ili ne postoji ili ima
        vrednost vecu ili jednaku od vrednosti novog cvora.*/
99
     /* Zbog izracunavanja izraza u C-u prvi deo konjunkcije mora biti
        provera da li se doslo do poslednjeg cvora liste pre nego sto
        se proveri vrednost u sledecem cvoru jer u slucaju poslednjeg,
        sledeci ne postoji pa ni njegova vrednost. */
     while (glava->sledeci != NULL && glava->sledeci->vrednost < broj)</pre>
       glava = glava->sledeci;
     /* Iz petlje se moglo izaci pomeranjem pokazivaca glava do
        poslednjeg cvora ili, ranije, nailaskom na cvor ciji sledeci
        ima vrednost vecu od broj */
     return glava;
111 }
int dodaj_iza(Cvor * tekuci, int broj)
     /* Kreira se novi cvor i provera se uspesnost kreiranja */
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
       return 1;
119
    novi->sledeci = tekuci->sledeci;
    novi->prethodni = tekuci;
```

```
/* Ako tekuci ima sledeceg, onda se sledecem dodeljuje prethodnik,
        a potom i tekuci dobija novog sledeceg postavljanjem pokazivaca
        na ispravne adrese */
     if (tekuci->sledeci != NULL)
       tekuci->sledeci->prethodni = novi;
     tekuci->sledeci = novi;
129
     /* Vraca se indikator uspesnog dodavanja */
    return 0;
   int dodaj_sortirano(Cvor ** adresa_glave, Cvor ** adresa_kraja, int
                       broj)
     /* Ako je lista prazna, novi cvor je i prvi i poslednji cvor */
     if (*adresa_glave == NULL) {
       /* Kreira se novi cvor i proverava se uspesnost kreiranja */
139
       Cvor *novi = napravi_cvor(broj);
       if (novi == NULL)
141
         return 1;
143
       /* Azuriraju se vrednosti pocetka i kraja liste */
       *adresa_glave = novi;
145
       *adresa_kraja = novi;
147
       /* Vraca se indikator uspesnog dodavanja */
       return 0;
149
     }
     /* Ukoliko je vrednost glave liste veca ili jednaka od nove
        vrednosti onda novi cvor treba staviti na pocetak liste */
     if ((*adresa_glave)->vrednost >= broj) {
      return dodaj_na_pocetak_liste(adresa_glave, adresa_kraja, broj);
     /* Pronazi se cvor iza koga treba uvezati novi cvor */
     Cvor *pomocni = pronadji_mesto_umetanja(*adresa_glave, broj);
159
     /* Dodaje se novi cvor uz proveru uspesnosti dodavanja */
     if (dodaj_iza(pomocni, broj) == 1)
      return 1;
     /* Ako pomocni cvor pokazuje na poslednji element liste, onda je
        novi cvor poslednji u listi. */
     if (pomocni == *adresa_kraja)
       *adresa_kraja = pomocni->sledeci;
167
     return 0;
  }
  Cvor *pretrazi_listu(Cvor * glava, int broj)
     /* Obilaze se cvorovi liste */
     for (; glava != NULL; glava = glava->sledeci)
```

```
/* Ako je vrednost tekuceg cvora jednaka zadatom broju, pretraga
          se obustavlja */
       if (glava->vrednost == broj)
         return glava;
179
     /* Nema trazenog broja u listi i vraca se NULL */
     return NULL;
181
183
   Cvor *pretrazi_sortiranu_listu(Cvor * glava, int broj)
185
     /* Obilaze se cvorovi liste */
     /* U uslovu ostanka u petlji, bitan je redosled u konjunkciji */
187
     for (; glava != NULL && glava->vrednost <= broj;</pre>
          glava = glava->sledeci)
189
       /* Ako je vrednost tekuceg cvora jednaka zadatom broju, pretraga
          se obustavlja */
191
       if (glava->vrednost == broj)
         return glava;
     /* Nema trazenog broja u listi i bice vraceno NULL */
195
     return NULL;
  ۱,
197
  /* Kod dvostruko povezane liste brisanje odredjenog cvora se moze
199
      lako realizovati jer on sadrzi pokazivace na svog sledbenika i
      prethodnika u listi. U funkciji se bise cvor zadat argumentom
201
      tekuci */
void obrisi_tekuci(Cvor ** adresa_glave, Cvor ** adresa_kraja, Cvor *
                      tekuci)
205
     /* Ako je tekuci NULL pokazivac, nema potrebe za brisanjem */
     if (tekuci == NULL)
207
       return:
209
     /* Ako postoji prethodnik tekuceg cvora, onda se postavlja da
        njegov sledbenik bude sledbenik tekuceg cvora */
211
     if (tekuci->prethodni != NULL)
       tekuci->prethodni->sledeci = tekuci->sledeci;
213
     /* Ako postoji sledbenik tekuceg cvora, onda njegov prethodnik
215
        treba da bude prethodnik tekuceg cvora */
     if (tekuci->sledeci != NULL)
217
       tekuci->sledeci->prethodni = tekuci->prethodni;
219
     /* Ako je glava cvor koji se brise, nova glava liste ce biti
        sledbenik stare glave */
221
     if (tekuci == *adresa_glave)
       *adresa_glave = tekuci->sledeci;
223
     /* Ako je cvor koji se brise poslednji u listi, azurira se i
        pokazivac na kraj liste */
```

```
if (tekuci == *adresa_kraja)
       *adresa_kraja = tekuci->prethodni;
     /* Oslobadja se dinamicki alociran prostor za cvor tekuci */
     free(tekuci);
231
   void obrisi_cvor(Cvor ** adresa_glave, Cvor ** adresa_kraja,
                    int broj)
     Cvor *tekuci = *adresa_glave;
     /* Sve dok ima cvorova cija je vrednost jednaka zadatom broju,
        takvi cvorovi se brisu iz liste. */
     while ((tekuci = pretrazi_listu(*adresa_glave, broj)) != NULL)
       obrisi_tekuci(adresa_glave, adresa_kraja, tekuci);
  }
  void obrisi_cvor_sortirane_liste(Cvor ** adresa_glave, Cvor **
                                     adresa_kraja, int broj)
247
     Cvor *tekuci = *adresa_glave;
249
     /* Sve dok ima cvorova cija je vrednost jednaka zadatom broju,
        takvi cvorovi se brisu iz liste. */
251
     while ((tekuci =
             pretrazi_sortiranu_listu(*adresa_glave, broj)) != NULL)
253
       obrisi_tekuci(adresa_glave, adresa_kraja, tekuci);
   }
   void ispisi_listu(Cvor * glava)
257
     putchar('[');
     /* Unutar zagrada ispisuju se vrednosti u cvorovima liste od
261
        pocetka prema kraju liste */
     for (; glava != NULL; glava = glava->sledeci) {
       printf("%d", glava->vrednost);
263
       if (glava->sledeci != NULL)
         printf(", ");
265
267
     printf("]\n");
   }
269
   void ispisi_listu_unazad(Cvor * kraj)
     putchar('[');
     /* Unutar zagrada ispisuju se vrednosti u cvorovima liste od kraja
        prema pocetku liste */
     for (; kraj != NULL; kraj = kraj->prethodni) {
       printf("%d", kraj->vrednost);
       if (kraj->prethodni != NULL)
```

```
printf(", ");
}
281  printf("]\n");
}
```

#### main a.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "dvostruko_povezana_lista.h"
  /* 1) Glavni program */
6 int main()
    /* Lista je prazna na pocetku */
    /* Cuvaju se pokazivaci na glavu liste i na poslednji cvor liste,
       da bi operacije poput dodavanja na kraj liste i ispisivanja
       liste unazad bile efikasne poput dodavanja na pocetak liste i
12
       ispisivanja liste od pocetnog do poslednjeg cvora. */
    Cvor *glava = NULL;
14
    Cvor *kraj = NULL;
    Cvor *trazeni = NULL;
    int broj;
    /* Testira se funkcija za dodavanje novog broja na pocetak liste */
18
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", &broj) > 0) {
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
         memorije za novi cvor. Memoriju alociranu za cvorove liste
         treba osloboditi pre napustanja programa */
      if (dodaj_na_pocetak_liste(&glava, &kraj, broj) == 1) {
24
        fprintf(stderr,
                "Greska: Neuspesna alokacija memorije za cvor.\n");
26
        oslobodi_listu(&glava, &kraj);
        exit(EXIT_FAILURE);
28
      printf("\tLista: ");
30
      ispisi_listu(glava);
34
    /* Testira se funkcija za pretragu liste */
    printf("\nUnesite broj koji se trazi u listi: ");
    scanf("%d", &broj);
36
    /* Pokazivac trazeni dobija vrednost rezultata pretrage */
38
    trazeni = pretrazi_listu(glava, broj);
    if (trazeni == NULL)
40
      printf("Broj %d se ne nalazi u listi!\n", broj);
42
      printf("Trazeni broj %d je u listi!\n", trazeni->vrednost);
44
```

```
/* Ispisuje se lista unazad */
printf("\nLista ispisana u nazad: ");
ispisi_listu_unazad(kraj);

/* Oslobadja se memorija zauzeta za cvorove liste */
oslobodi_listu(&glava, &kraj);

exit(EXIT_SUCCESS);
}
```

#### $main_b.c$

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include "dvostruko_povezana_lista.h"
5 /* 2) Glavni program */
  int main()
    /* Lista je prazna na pocetku. */
    Cvor *glava = NULL;
    Cvor *kraj = NULL;
    int broj;
    /* Testira se funkcija za dodavanje novog broja na kraj liste */
13
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", &broj) > 0) {
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
         memorije za novi cvor. Memoriju alociranu za cvorove liste
         treba osloboditi pre napustanja programa */
19
      if (dodaj_na_kraj_liste(&glava, &kraj, broj) == 1) {
        fprintf(stderr,
                 "Greska: Neuspesna alokacija memorije za cvor.\n");
        oslobodi_listu(&glava, &kraj);
        exit(EXIT_FAILURE);
      printf("\tLista: ");
      ispisi_listu(glava);
27
    /* Testira se funkcija za brisanje elemenata iz liste */
    printf("\nUnesite broj koji se brise iz liste: ");
    scanf("%d", &broj);
31
    /* Brisu se cvorovi iz liste cije polje vrednost je jednako broju
33
       procitanom sa ulaza */
    obrisi_cvor(&glava, &kraj, broj);
35
    printf("Lista nakon brisanja: ");
    ispisi_listu(glava);
39
```

```
/* Ispisuje se lista unazad */
printf("\nLista ispisana u nazad: ");
ispisi_listu_unazad(kraj);

/* Oslobadja se memorija zauzeta za cvorove liste */
oslobodi_listu(&glava, &kraj);

exit(EXIT_SUCCESS);
}
```

main\_c.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "dvostruko_povezana_lista.h"
  /* 3) Glavni program */
6 int main()
    /* Lista je prazna na pocetku */
    Cvor *glava = NULL;
    Cvor *kraj = NULL;
    Cvor *trazeni = NULL;
12
    int broj;
14
    /* Testira se funkcija za dodavanje vrednosti u listu tako da ona
       bude uredjena neopadajuce */
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", &broj) > 0) {
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
18
         memorije za novi cvor. Memoriju alociranu za cvorove liste
         treba osloboditi pre napustanja programa */
20
      if (dodaj_sortirano(&glava, &kraj, broj) == 1) {
        fprintf(stderr,
                "Greska: Neuspesna alokacija memorije za cvor.\n");
24
        oslobodi_listu(&glava, &kraj);
        exit(EXIT_FAILURE);
26
      printf("\tLista: ");
28
      ispisi_listu(glava);
30
    /* Testira se funkcija za pretragu liste */
    printf("\nUnesite broj koji se trazi u listi: ");
    scanf("%d", &broj);
34
    /* Pokazivac trazeni dobija vrednost rezultata pretrage */
    trazeni = pretrazi_listu(glava, broj);
36
    if (trazeni == NULL)
      printf("Broj %d se ne nalazi u listi!\n", broj);
38
    else
```

```
printf("Trazeni broj %d je u listi!\n", trazeni->vrednost);
40
    /* Testira se funkcija za brisanje elemenata iz liste */
42
    printf("\nUnesite broj koji se brise iz liste: ");
    scanf("%d", &broj);
44
    /* Brisu se cvorovi iz liste cije polje vrednost je jednako broju
46
       procitanom sa ulaza */
    obrisi_cvor_sortirane_liste(&glava, &kraj, broj);
48
    printf("Lista nakon brisanja: ");
    ispisi_listu(glava);
    /* Ispisuje se lista unazad */
    printf("\nLista ispisana u nazad: ");
54
    ispisi_listu_unazad(kraj);
56
    /* Oslobadja se memorija zauzeta za cvorove liste */
    oslobodi_listu(&glava, &kraj);
58
    exit(EXIT_SUCCESS);
```

#### stabla.h

```
#ifndef _STABLA_H_
2 #define _STABLA_H_ 1
  /* a) Struktura kojom se predstavlja cvor binarnog pretrazivackog
     stabla */
  typedef struct cvor {
   int broj;
   struct cvor *levo;
    struct cvor *desno;
10 } Cvor;
12 /* b) Funkcija koja alocira memoriju za novi cvor stabla,
     inicijalizuje polja strukture i vraca pokazivac na novi cvor */
14 Cvor *napravi_cvor(int broj);
16 /* c) Funkcija koja dodaje zadati broj u stablo. Povratna vrednost
     funkcije je 0 ako je dodavanje uspesno, odnosno 1 ukoliko je
    doslo do greske */
  int dodaj_u_stablo(Cvor ** adresa_korena, int broj);
  /* d) Funkcija koja proverava da li se zadati broj nalazi stablu */
22 Cvor *pretrazi_stablo(Cvor * koren, int broj);
```

```
24 /* e) Funkcija koja pronalazi cvor koji sadrzi najmanju vrednost u
    stablu */
26 | Cvor *pronadji_najmanji(Cvor * koren);
28 /* f) Funkcija koja pronalazi cvor koji sadrzi najvecu vrednost u
    stablu */
30 Cvor *pronadji_najveci(Cvor * koren);
32 /* g) Funkcija koja brise cvor stabla koji sadrzi zadati broj */
  void obrisi_element(Cvor ** adresa_korena, int broj);
  /* h) Funkcija koja ispisuje stablo u infiksnoj notaciji (Levo
    postablo - Koren - Desno podstablo ) */
36
  void ispisi_stablo_infiksno(Cvor * koren);
38
  /* i) Funkcija koja ispisuje stablo u prefiksnoj notaciji ( Koren -
    Levo podstablo - Desno podstablo ) */
40
  void ispisi_stablo_prefiksno(Cvor * koren);
42
  /* j) Funkcija koja ispisuje stablo u postfiksnoj notaciji ( Levo
    podstablo - Desno postablo - Koren) */
44
  void ispisi_stablo_postfiksno(Cvor * koren);
46
  /* k) Funkcija koja oslobadja memoriju zauzetu stablom */
48 void oslobodi_stablo(Cvor ** adresa_korena);
50 #endif
```

#### stabla.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "stabla.h"
  Cvor *napravi_cvor(int broj)
6
    /* Alocira se memorija za novi cvor i proverava se uspesnost
      alokacije. */
   Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
   if (novi == NULL)
     return NULL;
12
    /* Inicijalizuju se polja novog cvora. */
   novi->broj = broj;
14
   novi->levo = NULL;
   novi->desno = NULL;
16
    /* Vraca se adresa novog cvora. */
    return novi;
20 }
```

```
22 int dodaj_u_stablo(Cvor ** adresa_korena, int broj)
    /* Ako je stablo prazno */
24
    if (*adresa_korena == NULL) {
26
      /* Kreira se novi cvor */
      Cvor *novi_cvor = napravi_cvor(broj);
28
      /* Proverava se uspesnost kreiranja */
30
      if (novi_cvor == NULL) {
        /* Ukoliko je doslo do greske, vraca se odgovarajuca vrednost
        return 1;
34
      /* Inace ... */
36
      /* Novi cvor se proglasava korenom stabla */
      *adresa_korena = novi_cvor;
38
      /* I vraca se indikator uspesnosti kreiranja */
40
      return 0;
42
    /* Ako stablo nije prazno, trazi se odgovarajuca pozicija za
44
       zadati broj */
46
    /* Ako je zadata vrednost manja od vrednosti korena */
    if (broj < (*adresa_korena)->broj)
48
      /* Broj se dodaje u levo podstablo */
      return dodaj_u_stablo(&(*adresa_korena)->levo, broj);
    else
      /* Inace, broj je veci (ili jednak) od vrednosti u korenu pa se
54
         dodaje u desno podstablo */
      return dodaj_u_stablo(&(*adresa_korena)->desno, broj);
56
58
  Cvor *pretrazi_stablo(Cvor * koren, int broj)
60
    /* Ako je stablo prazno, vrednost se sigurno ne nalazi u njemu */
62
    if (koren == NULL)
      return NULL;
64
    /* Ako je trazena vrednost sadrzana u korenu */
66
    if (koren->broj == broj) {
68
      /* Prekida se pretraga */
      return koren;
70
72
```

```
/* Inace, ako je broj manji od vrednosti sadrzane u korenu */
     if (broj < koren->broj)
       /* Pretraga se nastavlja u levom podstablu */
       return pretrazi_stablo(koren->levo, broj);
       /* U suprotnom, pretraga se nastavlja u desnom podstablu */
80
       return pretrazi_stablo(koren->desno, broj);
  }
82
  Cvor *pronadji_najmanji(Cvor * koren)
84
86
     /* Ako je stablo prazno, prekida se pretraga */
     if (koren == NULL)
88
       return NULL;
90
     /* Vrednosti koje su manje od vrednosti u korenu stabla nalaze se
        levo od njega */
92
     /* Ako je koren cvor koji nema levo podstablo, onda on sadrzi
94
        najmanju vrednost */
     if (koren->levo == NULL)
96
      return koren;
98
     /* Inace, pretragu treba nastaviti u levom podstablu */
     return pronadji_najmanji(koren->levo);
   Cvor *pronadji_najveci(Cvor * koren)
104
     /* Ako je stablo prazno, prekida se pretraga */
     if (koren == NULL)
106
       return NULL;
108
     /* Vrednosti koje su vece od vrednosti u korenu stabla nalaze se
        desno od njega */
     /* Ako je koren cvor koji nema desno podstablo, onda on sadrzi
112
        najvecu vrednost */
     if (koren->desno == NULL)
114
       return koren;
116
     /* Inace, pretragu treba nastaviti u desnom podstablu */
     return pronadji_najveci(koren->desno);
118
120
   void obrisi_element(Cvor ** adresa_korena, int broj)
122 {
     Cvor *pomocni_cvor = NULL;
124
```

```
/* Ako je stablo prazno, brisanje nije primenljivo */
     if (*adresa korena == NULL)
126
       return:
128
     /* Ako je vrednost koju treba obrisati manja od vrednosti u korenu
        stabla, ona se eventualno nalazi u levom podstablu, pa treba
130
        rekurzivno primeniti postupak na levo podstablo. Koren ovako
        modifikovanog stabla je nepromenjen. */
     if (broj < (*adresa_korena)->broj) {
       obrisi_element(&(*adresa_korena)->levo, broj);
134
       return;
136
     /* Ako je vrednost koju treba obrisati veca od vrednosti u korenu
138
        stabla, ona se eventualno nalazi u desnom podstablu pa treba
        rekurzivno primeniti postupak na desno podstablo. Koren ovako
140
        modifikovanog stabla je nepromenjen. */
     if ((*adresa_korena)->broj < broj) {
142
       obrisi_element(&(*adresa_korena)->desno, broj);
       return;
144
146
     /* Slede podslucajevi vezani za slucaj kada je vrednost u korenu
        jednaka broju koji se brise (tj. slucaj kada treba obrisati
148
        koren) */
     /* Ako koren nema sinova, tada se on prosto brise, i rezultat je
        prazno stablo (vraca se NULL) */
     if ((*adresa_korena)->levo == NULL
         && (*adresa_korena)->desno == NULL) {
154
       free(*adresa_korena);
       *adresa_korena = NULL;
       return;
158
     /* Ako koren ima samo levog sina, tada se brisanje vrsi tako sto
        se brise koren, a novi koren postaje levi sin */
     if ((*adresa_korena)->levo != NULL
         && (*adresa_korena)->desno == NULL) {
       pomocni_cvor = (*adresa_korena)->levo;
164
       free(*adresa_korena);
       *adresa_korena = pomocni_cvor;
       return;
     }
168
     /* Ako koren ima samo desnog sina, tada se brisanje vrsi tako sto
        se brise koren, a novi koren postaje desni sin */
     if ((*adresa_korena)->desno != NULL
         && (*adresa_korena)->levo == NULL) {
       pomocni_cvor = (*adresa_korena)->desno;
       free(*adresa_korena);
       *adresa_korena = pomocni_cvor;
```

```
return:
178
     /* Slucaj kada koren ima oba sina - najpre se potrazi sledbenik
180
        korena (u smislu poretka) u stablu. To je upravo po vrednosti
        najmanji cvor u desnom podstablu. On se moze pronaci npr.
182
        funkcijom pronadji_najmanji(). Nakon toga se u koren smesti
        vrednost tog cvora, a u taj cvor se smesti vrednost korena (tj.
184
        broj koji se brise). Zatim se prosto rekurzivno pozove funkcija
        za brisanje na desno podstablo. S obzirom da u njemu treba
186
        obrisati najmanji element, a on zasigurno ima najvise jednog
        potomka, jasno je da ce njegovo brisanje biti obavljeno na jedan
188
        od jednostavnijih nacina koji su gore opisani. */
     pomocni_cvor = pronadji_najmanji((*adresa_korena)->desno);
190
     (*adresa_korena)->broj = pomocni_cvor->broj;
     pomocni_cvor->broj = broj;
     obrisi_element(&(*adresa_korena)->desno, broj);
194
void ispisi_stablo_infiksno(Cvor * koren)
     /* Ako stablo nije prazno */
198
     if (koren != NULL) {
200
       /* Prvo se ispisuju svi cvorovi levo od korena */
       ispisi_stablo_infiksno(koren->levo);
202
       /* Zatim se ispisuje vrednost u korenu */
204
       printf("%d ", koren->broj);
206
       /* Na kraju se ispisuju cvorovi desno od korena */
       ispisi_stablo_infiksno(koren->desno);
208
  }
void ispisi_stablo_prefiksno(Cvor * koren)
     /* Ako stablo nije prazno */
214
     if (koren != NULL) {
       /* Prvo se ispisuje vrednost u korenu */
       printf("%d ", koren->broj);
218
       /* Zatim se ispisuju svi cvorovi levo od korena */
       ispisi_stablo_prefiksno(koren->levo);
       /* Na kraju se ispisuju svi cvorovi desno od korena */
       ispisi_stablo_prefiksno(koren->desno);
  }
226
void ispisi_stablo_postfiksno(Cvor * koren)
```

```
230
     /* Ako stablo nije prazno */
     if (koren != NULL) {
       /* Prvo se ispisuju svi cvorovi levo od korena */
234
       ispisi_stablo_postfiksno(koren->levo);
236
       /* Zatim se ispisuju svi cvorovi desno od korena */
       ispisi_stablo_postfiksno(koren->desno);
238
       /* Na kraju se ispisuje vrednost u korenu */
240
       printf("%d ", koren->broj);
242
244
   void oslobodi_stablo(Cvor ** adresa_korena)
246
     /* Ako je stablo prazno, nepotrebno je oslobadjati memoriju */
     if (*adresa_korena == NULL)
248
       return;
     /* Inace ... */
     /* Oslobadja se memorija zauzeta levim podstablom */
     oslobodi_stablo(&(*adresa_korena)->levo);
254
     /* Oslobadja se memorija zauzetu desnim podstablom */
     oslobodi_stablo(&(*adresa_korena)->desno);
     /* Oslobadja se memorija zauzeta korenom */
258
     free(*adresa_korena);
260
     /* Proglasava se stablo praznim */
     *adresa_korena = NULL;
262
```

#### main.c

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

int main()
{
    Cvor *koren;
    int n;
    Cvor *trazeni_cvor;

/* Proglasava se stablo praznim */
    koren = NULL;
/*

#include <stdio.h>
#include <stdio.h

#include <stdio.h>
#include <stdio.h

#include
```

```
/* Citaju se vrednosti i dodaju u stablo uz proveru uspesnosti
       dodavanja */
    printf("Unesite brojeve (CTRL+D za kraj unosa): ");
    while (scanf("%d", &n) != EOF) {
17
      if (dodaj_u_stablo(&koren, n) == 1) {
        fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n", n);
19
        oslobodi stablo(&koren);
        exit(EXIT_FAILURE);
    }
23
    /* Generisu se trazeni ispisi: */
    printf("\nInfiksni ispis: ");
    ispisi_stablo_infiksno(koren);
    printf("\nPrefiksni ispis: ");
    ispisi_stablo_prefiksno(koren);
    printf("\nPostfiksni ispis: ");
    ispisi_stablo_postfiksno(koren);
    /* Demonstrira se rad funkcije za pretragu */
33
    printf("\nTrazi se broj: ");
    scanf("%d", &n);
35
    trazeni_cvor = pretrazi_stablo(koren, n);
    if (trazeni_cvor == NULL)
      printf("Broj se ne nalazi u stablu!\n");
      printf("Broj se nalazi u stablu!\n");
41
    /* Demonstrira se rad funkcije za brisanje */
43
    printf("Brise se broj: ");
    scanf("%d", &n);
45
    obrisi_element(&koren, n);
    printf("Rezultujuce stablo: ");
47
    ispisi_stablo_infiksno(koren);
    printf("\n");
49
    /* Oslobadja se memorija zauzeta stablom */
    oslobodi_stablo(&koren);
53
    exit(EXIT_SUCCESS);
  }
55
```

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

#define MAX 50
```

```
/* Struktura kojom se opisuje cvor stabla: sadrzi rec, njen broj
     pojavljivanja i redom pokazivace na levo i desno podstablo */
10 typedef struct cvor {
    char *rec:
    int brojac;
    struct cvor *levo;
    struct cvor *desno;
  } Cvor;
16
  /* Funkcija koja kreira novi cvora stabla */
18 Cvor *napravi_cvor(char *rec)
    /* Alocira se memorija za novi cvor i proverava se uspesnost
20
       alokacije. */
    Cvor *novi_cvor = (Cvor *) malloc(sizeof(Cvor));
    if (novi_cvor == NULL)
      return NULL:
24
    /* Alocira se memorija za zadatu rec: potrebno je rezervisati
26
       memoriju za svaki karakter reci ukljucujuci i terminirajucu
       nulu */
28
    novi_cvor->rec = (char *) malloc((strlen(rec) + 1) * sizeof(char));
    if (novi_cvor->rec == NULL) {
30
      free(novi cvor);
      return NULL;
34
    /* Inicijalizuju se polja u novom cvoru */
    strcpy(novi_cvor->rec, rec);
36
    novi_cvor->brojac = 1;
    novi_cvor->levo = NULL;
38
    novi_cvor->desno = NULL;
40
    /* Vraca se adresa novog cvora */
42
    return novi_cvor;
44
  /* Funkcija koja dodaje novu rec u stablo - ukoliko je dodavanje
     uspesno povratna vrednost je 0, u suprotnom povratna vrednost je
46
  int dodaj_u_stablo(Cvor ** adresa_korena, char *rec)
48
    /* Ako je stablo prazno */
    if (*adresa_korena == NULL) {
      /* Kreira se cvor koji sadrzi zadatu rec */
      Cvor *novi_cvor = napravi_cvor(rec);
      /* Proverava se uspesnost kreiranja novog cvora */
      if (novi_cvor == NULL) {
        /* I ukoliko je doslo do greske, vraca se odgovarajuca
56
           vrednost */
        return 1;
```

```
/* Inace... */
      /* Novi cvor se proglasava korenom stabla */
      *adresa_korena = novi_cvor;
      /* I vraca se indikator uspesnog dodavanja */
64
      return 0;
     /* Ako stablo nije prazno, trazi se odgovarajuca pozicija za novu
68
       rec */
     /* Ako je rec leksikografski manja od reci u korenu ubacuje se u
       levo podstablo */
     if (strcmp(rec, (*adresa_korena)->rec) < 0)</pre>
      return dodaj_u_stablo(&(*adresa_korena)->levo, rec);
74
    else
       /* Ako je rec leksikografski veca od reci u korenu ubacuje se u
         desno podstablo */
78
     if (strcmp(rec, (*adresa_korena)->rec) > 0)
      return dodaj_u_stablo(&(*adresa_korena)->desno, rec);
80
    else {
82
      /* Ako je rec jednaka reci u korenu, uvecava se njen broj
         pojavljivanja */
84
      (*adresa_korena)->brojac++;
86
      /* I vraca se indikator uspesnog dodavanja */
      return 0;
88
     7
  }
90
92 /* Funkcija koja oslobadja memoriju zauzetu stablom */
   void oslobodi_stablo(Cvor ** adresa_korena)
94 {
     /* Ako je stablo prazno, nepotrebno je oslobadjati memoriju */
    if (*adresa_korena == NULL)
96
      return;
98
     /* Inace ... */
     /* Oslobadja se memorija zauzeta levim podstablom */
     oslobodi_stablo(&(*adresa_korena)->levo);
     /* Oslobadja se memorija zauzeta desnim podstablom */
     oslobodi_stablo(&(*adresa_korena)->desno);
104
     /* Oslobadja se memorija zauzeta korenom */
106
     free((*adresa_korena)->rec);
    free(*adresa_korena);
108
    /* Stablo se proglasava praznim */
```

```
*adresa_korena = NULL;
  }
112
   /* Funkcija koja pronalazi cvor koji sadrzi najfrekventniju rec (rec
      sa najvecim brojem pojavljivanja) */
  Cvor *nadji_najfrekventniju_rec(Cvor * koren)
116
     Cvor *max, *max_levo, *max_desno;
118
     /* Ako je stablo prazno, prekida se sa pretragom */
120
     if (koren == NULL)
       return NULL;
     /* Pronalazi se najfrekventnija rec u levom podstablu */
124
     max_levo = nadji_najfrekventniju_rec(koren->levo);
126
     /* Pronalazi se najfrekventnija rec u desnom podstablu */
     max_desno = nadji_najfrekventniju_rec(koren->desno);
128
     /* Trazi se maksimum vrednosti pojavljivanja reci iz levog
130
        podstabla, korena i desnog podstabla */
     max = koren;
     if (max_levo != NULL && max_levo->brojac > max->brojac)
       max = max_levo;
134
     if (max_desno != NULL && max_desno->brojac > max->brojac)
       max = max_desno;
136
     /* Vraca se adresa cvora sa najvecim brojem pojavljivanja */
     return max;
  }
140
   /* Funkcija koja ispisuje reci iz stabla u leksikografskom poretku
      pracene brojem pojavljivanja */
   void prikazi_stablo(Cvor * koren)
144
     /* Ako je stablo prazno, zavrsava se sa ispisom */
146
     if (koren == NULL)
       return;
148
     /* Zbog leksikografskog poretka, prvo se ispisuju sve reci iz
        levog podstabla */
     prikazi_stablo(koren->levo);
     /* Zatim rec iz korena */
     printf("%s: %d\n", koren->rec, koren->brojac);
     /* I nastavlja se sa ispisom reci iz desnog podstabla */
     prikazi_stablo(koren->desno);
158
160
   /* Funkcija ucitava sledecu rec iz zadate datoteke f i upisuje je u
      niz rec. Maksimalna duzina reci je odredjena argumentom max.
```

```
Funkcija vraca EOF ako u datoteci nema vise reci ili 0 u
     suprotnom. Rec je niz malih ili velikih slova. */
   int procitaj_rec(FILE * f, char rec[], int max)
166 {
     /* Karakter koji se cita */
     int c;
168
     /* Indeks pozicije na koju se smesta procitani karakter */
     int i = 0;
     /* Sve dok ima mesta za jos jedan karakter u nizu i dokle se god
       nije stiglo do kraja datoteke... */
174
     while (i < max - 1 && (c = fgetc(f)) != EOF) {
       /* Proverava se da li je procitani karakter slovo */
       if (isalpha(c))
         /* Ako jeste, smesta se u niz - pritom se vrsi konverzija u
178
            mala slova jer program treba da bude neosetljiv na razliku
            izmedju malih i velikih slova */
180
         rec[i++] = tolower(c);
182
       else
         /* Ako nije, proverava se da li je procitano barem jedno slovo
184
            nove reci */
         /* Ako jeste, prekida se sa citanjem */
186
       if (i > 0)
         break:
188
       /* U suprotnom se ide na sledecu iteraciju */
190
192
     /* Dodaje se na rec terminirajuca nula */
     rec[i] = '\0';
194
     /* Vraca se 0 ako je procitana rec, tj. EOF u suprotnom */
     return i > 0 ? 0 : EOF;
198 }
200 int main(int argc, char **argv)
     Cvor *koren = NULL, *max;
202
    FILE *f;
     char rec[MAX]:
204
     /* Provera da li je navedeno ime datoteke prilikom pokretanja
        programa */
     if (argc < 2) {
208
      fprintf(stderr, "Greska: Nedostaje ime ulazne datoteke!\n");
       exit(EXIT_FAILURE);
210
     7
     /* Priprema datoteke za citanje */
     if ((f = fopen(argv[1], "r")) == NULL) {
214
```

```
fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s.\n",
               argv[1]);
       exit(EXIT_FAILURE);
218
     /* Ucitavanje reci iz datoteke i smestanje u binarno stablo
        pretrage uz proveru uspesnosti dodavanja */
     while (procitaj_rec(f, rec, MAX) != EOF) {
       if (dodaj_u_stablo(&koren, rec) == 1) {
         fprintf(stderr, "Greska: Neuspelo dodavanje reci %s.\n", rec);
224
         oslobodi_stablo(&koren);
         exit(EXIT_FAILURE);
226
     }
228
     /* Posto je citanje reci zavrseno, zatvara se datoteka */
230
     fclose(f);
     /* Prikazuju se sve reci iz teksta i brojevi njihovih
        pojavljivanja. */
234
     prikazi_stablo(koren);
236
     /* Pronalazi se najfrekventnija rec */
     max = nadji_najfrekventniju_rec(koren);
238
     /* Ako takve reci nema... */
240
     if (max == NULL)
       /* Ispisuje se odgovarajuce obavestenje */
       printf("U tekstu nema reci!\n");
244
     else
246
       /* Inace, ispisuje se broj pojavljivanja reci */
       printf("Najcesca rec: %s (pojavljuje se %d puta)\n",
248
              max->rec, max->brojac);
     /* Oslobadja se dinamicki alociran prostor za stablo */
     oslobodi_stablo(&koren);
252
     exit(EXIT_SUCCESS);
254
```

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

#include <string.h>
#include <ctype.h>

#define MAX_IME_DATOTEKE 50
#define MAX_CIFARA 13
```

```
#define MAX_IME_I_PREZIME 100
  /* Struktura kojom se opisuje cvor stabla: sadrzi ime i prezime,
    broj telefona i redom pokazivace na levo i desno podstablo */
  typedef struct cvor {
    char ime_i_prezime[MAX_IME_I_PREZIME];
    char telefon[MAX_CIFARA];
   struct cvor *levo;
    struct cvor *desno;
17 } Cvor;
19 /* Funkcija koja kreira novi cvora stabla */
  Cvor *napravi_cvor(char *ime_i_prezime, char *telefon)
21 {
    /* Alocira se memorija za novi cvor i proverava se uspesnost
       alokacije. */
    Cvor *novi_cvor = (Cvor *) malloc(sizeof(Cvor));
    if (novi_cvor == NULL)
      return NULL;
    /* Inicijalizuju se polja novog cvora */
    strcpy(novi_cvor->ime_i_prezime, ime_i_prezime);
29
    strcpy(novi_cvor->telefon, telefon);
    novi_cvor->levo = NULL;
    novi_cvor->desno = NULL;
    /* Vraca se adresa novog cvora */
    return novi_cvor;
35
37
  /* Funkcija koja dodaje novu osobu i njen broj telefona u stablo -
     ukoliko je dodavanje uspesno povratna vrednost je 0, u suprotnom
39
     povratna vrednost je 1 */
  int.
41
  dodaj_u_stablo(Cvor ** adresa_korena, char *ime_i_prezime,
                 char *telefon)
43
    /* Ako je stablo prazno */
45
    if (*adresa_korena == NULL) {
      /* Kreira se novi cvor */
47
      Cvor *novi_cvor = napravi_cvor(ime_i_prezime, telefon);
      /* Proverava se uspesnost kreiranja novog cvora */
49
      if (novi_cvor == NULL) {
        /* I ukoliko je doslo do greske, vraca se odgovarajuca
           vrednost */
        return 1;
      }
      /* Inace... */
      /* Novi cvor se proglasava korenom stabla */
      *adresa_korena = novi_cvor;
      /* I vraca se indikator uspesnog dodavanja */
```

```
return 0;
61
    /* Ako stablo nije prazno, trazi se odgovarajuca pozicija za novi
       unos. Kako pretragu treba vrsiti po imenu i prezimenu, stablo
       treba da bude pretrazivacko po ovom polju */
    /* Ako je zadato ime i prezime leksikografski manje od imena i
67
       prezimena koje se nalazi u korenu, podaci se dodaju u levo
       podstablo */
    if (strcmp(ime_i_prezime, (*adresa_korena)->ime_i_prezime)
        < 0)
      return dodaj_u_stablo(&(*adresa_korena)->levo, ime_i_prezime,
                             telefon);
    else
      /* Ako je zadato ime i prezime leksikografski vece od imena i
         prezimena sadrzanog u korenu, podaci se dodaju u desno
         podstablo */
    if (strcmp(ime_i_prezime, (*adresa_korena)->ime_i_prezime) > 0)
79
      return dodaj_u_stablo(&(*adresa_korena)->desno, ime_i_prezime,
                             telefon);
81
    /* Pretostavka zadatka je da nema istih imena i prezimena u
83
       datoteci, pa se sledeca naredba nikada i neci izvrsiti */
    return 0;
85
87
  /* Funkcija koja oslobadja memoriju zauzetu stablom */
  void oslobodi_stablo(Cvor ** adresa_korena)
89
    /* Ako je stablo prazno, nepotrebno je oslobadjati memoriju */
91
    if (*adresa_korena == NULL)
      return:
93
    /* Inace ... */
95
    /* Oslobadja se memorija zauzeta levim podstablom */
    oslobodi_stablo(&(*adresa_korena)->levo);
97
    /* Oslobadja se memorija zauzeta desnim podstablom */
99
    oslobodi_stablo(&(*adresa_korena)->desno);
    /* Oslobadja se memorija zauzeta korenom */
    free(*adresa_korena);
    /* Stablo se proglasava praznim */
    *adresa_korena = NULL;
  /* Funkcija koja ispisuje imenik u leksikografskom poretku */
  /* Napomena: ova funkcija nije trazena u zadatku ali se moze
     koristiti za proveru da li je stablo lepo kreirano ili ne */
```

```
void prikazi_stablo(Cvor * koren)
113 {
     /* Ako je stablo prazno, zavrsava se sa ispisom */
     if (koren == NULL)
      return:
     /* Zbog leksikografskog poretka, prvo se ispisuju podaci iz levog
        podstabla */
119
     prikazi_stablo(koren->levo);
     /* Zatim se ispisuju podaci iz korena */
     printf("%s: %s\n", koren->ime_i_prezime, koren->telefon);
123
     /* I nastavlja se sa ispisom podataka iz desnog podstabla */
     prikazi_stablo(koren->desno);
  }
129 /* Funkcija ucitava sledeci kontakt iz zadate datoteke i upisuje ime
      i prezime i broj telefona u odgovarajuce nizove. Maksimalna duzina
      imena i prezimena odredjena je konstantom MAX_IME_PREZIME, a
      maksimalna duzina broja telefona konstantom MAX_CIFARA. Funkcija
     vraca EOF ako nema vise kontakata ili 0 u suprotnom. */
   int procitaj_kontakt(FILE * f, char *ime_i_prezime, char *telefon)
135 {
     /* Karakter koji se cita */
    int c:
     /* Indeks pozicije na koju se smesta procitani karakter */
139
     int i = 0;
     /* Linije datoteke koje se obradjuju su formata Ime Prezime
        BrojTelefona */
143
     /* Preskacu se eventualne praznine sa pocetka linije datoteke */
     while ((c = fgetc(f)) != EOF && isspace(c));
147
     /* Prvo procitano slovo upisuje se u ime i prezime */
     if (!feof(f))
       ime_i_prezime[i++] = c;
     /* Naznaka kraja citanja imena i prezimena ce biti pojava prve
        cifre tako da se citanje vrsi sve dok se ne naidje na cifru.
        Pritom treba voditi racuna da li ima dovoljno mesta za
        smestanje procitanog karaktera i da se slucajno ne dodje do
        kraja datoteke */
     while (i < MAX_IME_I_PREZIME - 1 && (c = fgetc(f)) != EOF) {
       if (!isdigit(c))
         ime_i_prezime[i++] = c;
       else if (i > 0)
161
         break;
163
     }
```

```
/* Upisuje se terminirajuca nula na mesto poslednjeg procitanog
165
        blanko karaktera */
     ime_i_prezime[--i] = '\0';
     /* I pocinje se sa citanjem broja telefona */
     i = 0;
     /* Upisuje se cifra koja je vec procitana */
     telefon[i++] = c;
173
     /* I citaju se preostale cifre. Naznaka kraja ce biti pojava
        karaktera cije prisustvo nije dozvoljeno u broju telefona */
     while (i < MAX_CIFARA - 1 && (c = fgetc(f)) != EOF)
       if (c == '/' || c == '-' || isdigit(c))
         telefon[i++] = c;
       else
         break:
181
     /* Upisuje se terminirajuca nula */
183
     telefon[i] = '\0';
185
     /* Vraca se 0 ako je procitan kontakt ili EOF u suprotnom */
     return !feof(f) ? 0 : EOF;
187
180
   /* Funkcija koja trazi u imeniku osobu sa zadatim imenom i
      prezimenom */
191
   Cvor *pretrazi_imenik(Cvor * koren, char *ime_i_prezime)
193
     /* Ako je imenik prazan, zavrsava se sa pretragom */
     if (koren == NULL)
195
       return NULL;
     /* Ako je trazeno ime i prezime sadrzano u korenu, takodje se
        zavrsava sa pretragom */
199
     if (strcmp(koren->ime_i_prezime, ime_i_prezime) == 0)
       return koren;
201
     /* Ako je zadato ime i prezime leksikografski manje od vrednosti u
203
        korenu pretraga se nastavlja levo */
     if (strcmp(ime_i_prezime, koren->ime_i_prezime) < 0)</pre>
205
       return pretrazi_imenik(koren->levo, ime_i_prezime);
207
     else
       /* U suprotnom, pretraga se nastavlja desno */
209
       return pretrazi_imenik(koren->desno, ime_i_prezime);
211
213 int main(int argc, char **argv)
     char ime_datoteke[MAX_IME_DATOTEKE];
```

```
Cvor *koren = NULL;
     Cvor *trazeni;
217
     FILE *f:
     char ime_i_prezime[MAX_IME_I_PREZIME];
219
     char telefon[MAX_CIFARA];
     char c:
     int i;
223
     /* Ucitava se ime datoteke i vrsi se njena priprema za citanje */
     printf("Unesite ime datoteke: ");
     scanf("%s", ime_datoteke);
     getchar();
     if ((f = fopen(ime_datoteke, "r")) == NULL) {
       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s.\n",
               ime_datoteke);
       exit(EXIT_FAILURE);
     /* Citaju se podaci iz datoteke i smestanju u binarno stablo
        pretrage uz proveru uspesnosti dodavanja */
     while (procitaj_kontakt(f, ime_i_prezime, telefon) != EOF)
       if (dodaj_u_stablo(&koren, ime_i_prezime, telefon) == 1) {
         fprintf(stderr,
                 "Greska: Neuspelo dodavanje podataka za osobu %s.\n",
                 ime_i_prezime);
         oslobodi_stablo(&koren);
         exit(EXIT_FAILURE);
     /* Zatvara se datoteka */
     fclose(f);
247
     /* Omogucava se pretraga imenika */
     while (1) {
       /* Ucitava se ime i prezime */
       printf("Unesite ime i prezime: ");
251
       i = 0;
       while ((c = getchar()) != '\n')
253
         ime_i_prezime[i++] = c;
       ime_i_prezime[i] = '\0';
       /* Ako je korisnik uneo naznaku za kraj pretrage, obustavlja se
257
          funkcionalnost */
       if (strcmp(ime_i_prezime, "KRAJ") == 0)
         break;
261
       /* Inace se ispisuje rezultat pretrage */
       trazeni = pretrazi_imenik(koren, ime_i_prezime);
263
       if (trazeni == NULL)
         printf("Broj nije u imeniku!\n");
265
       else
267
         printf("Broj je: %s \n", trazeni->telefon);
```

```
/* Oslobadja se memorija zauzeta imenikom */
oslobodi_stablo(&koren);

exit(EXIT_SUCCESS);
}
```

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
  #define MAX 51
  /* Struktura koja definise cvorove stabla: sadrzi ime i prezime
     studenta, ukupan uspeh, uspeh iz matematike, uspeh iz maternjeg
     jezika i redom pokazivace na levo i desno podstablo */
  typedef struct cvor_stabla {
    char ime[MAX];
    char prezime[MAX];
    double uspeh;
    double matematika;
    double jezik;
    struct cvor_stabla *levo;
    struct cvor_stabla *desno;
  } Cvor;
  /* Funkcija kojom se kreira cvor stabla */
21 Cvor *napravi_cvor(char ime[], char prezime[], double uspeh,
                      double matematika, double jezik)
    /* Alocira se memorija za novi cvor i proverava se uspesnost
       alokacije. */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
      return NULL;
29
    /* Inicijalizuju se polja strukture */
31
    strcpy(novi->ime, ime);
    strcpy(novi->prezime, prezime);
    novi->uspeh = uspeh;
33
    novi->matematika = matematika;
    novi->jezik = jezik;
    novi->levo = NULL;
    novi->desno = NULL;
    /* Vraca se adresa kreiranog cvora */
    return novi;
41 }
```

```
/* Funkcija koja dodaje cvor sa zadatim vrednostima u stablo -
     ukoliko je dodavanje uspesno povratna vrednost je 0, u suprotnom
     povratna vrednost je 1 */
45
  int dodaj_u_stablo(Cvor ** koren, char ime[], char prezime[],
                     double uspeh, double matematika, double jezik)
47
    /* Ako je stablo prazno */
49
    if (*koren == NULL) {
      /* Kreira se novi cvor */
      Cvor *novi_cvor =
          napravi_cvor(ime, prezime, uspeh, matematika, jezik);
53
      /* Proverava se uspesnost kreiranja novog cvora */
      if (novi_cvor == NULL) {
        /* I ukoliko je doslo do greske, vraca se odgovarajuca
           vrednost */
        return 1;
      /* Inace... */
      /* Novi cvor se proglasava korenom stabla */
      *koren = novi_cvor;
      /* I vraca se indikator uspesnog dodavanja */
      return 0;
    /* Ako stablo nije prazno, dodaje se cvor u stablo tako da bude
       sortirano po ukupnom broju poena */
    if (uspeh + matematika + jezik >
        (*koren)->uspeh + (*koren)->matematika + (*koren)->jezik)
      return dodaj_u_stablo(&(*koren)->levo, ime, prezime, uspeh,
                            matematika, jezik);
73
      return dodaj_u_stablo(&(*koren)->desno, ime, prezime, uspeh,
                            matematika, jezik);
  }
79
  /* Funkcija kojom se oslobadja memorija zauzeta stablom */
void oslobodi_stablo(Cvor ** koren)
    /* Ako je stablo prazno, nepotrebno je oslobadjati memoriju */
83
    if (*koren == NULL)
      return;
85
    /* Inace ... */
    /* Oslobadja se memorija zauzeta levim podstablom */
    oslobodi_stablo(&(*koren)->levo);
    /* Oslobadja se memorija zauzeta desnim podstablom */
91
    oslobodi_stablo(&(*koren)->desno);
93
```

```
/* Oslobadja se memorija zauzeta korenom */
     free(*koren);
95
     /* Stablo se proglasava praznim */
     *koren = NULL:
99
   /* Funkcija ispisuje sadrzaj stabla. Ukoliko je vrednost argumenta
      polozili jednaka O ispisuju se informacije o ucenicima koji nisu
      polozili prijemni, a ako je vrednost argumenta razlicita od nule,
      ispisuju se informacije o ucenicima koji su polozili prijemni */
   void stampaj(Cvor * koren, int polozili)
     /* Stablo je prazno - prekida se sa ispisom */
     if (koren == NULL)
       return;
     /* Stampaju se informacije iz levog podstabla */
     stampaj(koren->levo, polozili);
113
     /* Stampaju se informacije iz korenog cvora */
     if (polozili && koren->matematika + koren->jezik >= 10)
       printf("%s %s %.1lf %.1lf %.1lf %.1lf\n", koren->ime,
              koren->prezime, koren->uspeh, koren->matematika,
              koren->jezik,
119
              koren->uspeh + koren->matematika + koren->jezik);
     else if (!polozili && koren->matematika + koren->jezik < 10)
       printf("%s %s %.1lf %.1lf %.1lf \".1lf \n", koren->ime,
              koren->prezime, koren->uspeh, koren->matematika,
123
              koren->jezik,
              koren->uspeh + koren->matematika + koren->jezik);
     /* Stampaju se informacije iz desnog podstabla */
     stampaj(koren->desno, polozili);
   /* Funkcija koja odredjuje koliko studenata nije polozilo prijemni
      ispit */
   int nisu_polozili(Cvor * koren)
     /* Ako je stablo prazno, broj onih koji nisu polozili je 0 */
     if (koren == NULL)
       return 0;
     /* Pretraga se vrsi i u levom i u desnom podstablu - ako uslov za
139
        polaganje nije ispunjen za koreni cvor, broj studenata se
        uvecava za 1 */
141
     if (koren->matematika + koren->jezik < 10)
       return 1 + nisu_polozili(koren->levo) +
143
           nisu_polozili(koren->desno);
```

```
return nisu_polozili(koren->levo) + nisu_polozili(koren->desno);
147 }
149 int main(int argc, char **argv)
    FILE *in:
    Cvor *koren;
    char ime[MAX], prezime[MAX];
    double uspeh, matematika, jezik;
     /* Otvara se datoteke sa rezultatima sa prijemnog za citanje */
     in = fopen("prijemni.txt", "r");
157
     if (in == NULL) {
      fprintf(stderr,
               "Greska: Neuspesno otvaranje datoteke prijemni.txt.\n");
      exit(EXIT_FAILURE);
161
163
     /* Citanje podataka i dodavanje u stablo uz proveru uspesnosti
       dodavanja */
165
     koren = NULL:
     while (fscanf(in, "%s %s %lf %lf %lf", ime, prezime, &uspeh,
167
                  &matematika, &jezik) != EOF) {
       if (dodaj_u_stablo
           (&koren, ime, prezime, uspeh, matematika, jezik)
           == 1) {
         fprintf(stderr,
                 "Greska: Neuspelo dodavanje podataka za %s %s.\n", ime,
                 prezime);
         oslobodi_stablo(&koren);
         exit(EXIT_FAILURE);
       }
177
179
     /* Zatvaranje datoteke */
     fclose(in);
181
     /* Stampaju se prvo podaci o ucenicima koji su polozili prijemni */
183
     stampaj(koren, 1);
185
     /* Linija se iscrtava samo ako postoje ucenici koji nisu polozili
       prijemni */
187
     if (nisu_polozili(koren) != 0)
      printf("----\n"):
189
     /* Stampaju se podaci o ucenicima koji nisu polozili prijemni */
191
     stampaj(koren, 0);
193
     /* Oslobadja se memorija zauzeta stablom */
     oslobodi_stablo(&koren);
195
     exit(EXIT_SUCCESS);
197
```

}

```
1 #include <stdio.h>
  #include <stdlib.h>
3 #include <string.h>
5 #define MAX_NISKA 51
  /* Struktura koja opisuje jedan cvor stabla: sadrzi ime i prezime
     osobe, dan i mesec rodjenja i redom pokazivace na levo i desno
     podstablo */
  typedef struct cvor_stabla {
    char ime[MAX_NISKA];
    char prezime[MAX_NISKA];
    int dan;
    int mesec;
    struct cvor_stabla *levo;
    struct cvor_stabla *desno;
17 } Cvor;
19 /* Funkcija koja kreira novi cvor */
  Cvor *napravi_cvor(char ime[], char prezime[], int dan, int mesec)
    /* Alocira se memorija */
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
      return NULL;
    /* Inicijalizuju se polja strukture */
    strcpy(novi->ime, ime);
    strcpy(novi->prezime, prezime);
    novi->dan = dan;
    novi->mesec = mesec;
    novi->levo = NULL;
    novi->desno = NULL;
    /* Vraca se adresa novog cvora */
    return novi;
37 }
  /* Funkcija koja dodaje novi cvor u stablo. Stablo treba da bude
     uredjeno po datumu - prvo po mesecu, a zatim po danu. Ukoliko je
     dodavanje uspesno povratna vrednost je 0, u suprotnom povratna
     vrednost je 1 */
43 int dodaj_u_stablo(Cvor ** koren, char ime[], char prezime[],
                      int dan, int mesec)
    /* Ako je stablo prazno */
    if (*koren == NULL) {
```

```
/* Kreira se novi cvor */
49
      Cvor *novi_cvor = napravi_cvor(ime, prezime, dan, mesec);
      /* Proverava se uspesnost kreiranja novog cvora */
      if (novi_cvor == NULL) {
        /* I ukoliko je doslo do greske, vraca se odgovarajuca
           vrednost */
        return 1;
      /* Inace... Novi cvor se proglasava korenom stabla */
      *koren = novi_cvor;
59
      /* I vraca se indikator uspesnog dodavanja */
      return 0;
    /* Stablo se uredjuje po mesecu, a zatim po danu u okviru istog
       meseca */
    if (mesec < (*koren)->mesec)
     return dodaj_u_stablo(&(*koren)->levo, ime, prezime, dan, mesec);
    else if (mesec == (*koren)->mesec && dan < (*koren)->dan)
      return dodaj_u_stablo(&(*koren)->levo, ime, prezime, dan, mesec);
    else
      return dodaj_u_stablo(&(*koren)->desno, ime, prezime, dan,
                            mesec);
73 }
  /* Funkcija vrsi pretragu stabla i vraca cvor sa trazenim datumom */
  Cvor *pretrazi(Cvor * koren, int dan, int mesec)
    /* Stablo je prazno, obustavlja se pretraga */
    if (koren == NULL)
79
     return NULL;
81
    /* Ako je trazeni datum u korenu */
    if (koren->dan == dan && koren->mesec == mesec)
83
      return koren;
85
    /* Ako je mesec trazenog datuma manji od meseca sadrzanog u korenu
       ili ako su meseci isti ali je dan trazenog datuma manji od
87
       aktuelnog datuma, pretrazuje se levo podstablo - pre toga se
       svakako proverava da li leva grana postoji - ako ne postoji
89
       treba vratiti prvi sledeci, a to je bas vrednost uocenog korena
    if (mesec < koren->mesec
91
        || (mesec == koren->mesec && dan < koren->dan)) {
      if (koren->levo == NULL)
        return koren;
      else
95
        return pretrazi(koren->levo, dan, mesec);
    }
97
```

```
/* Inace se nastavlja pretraga u desnom delu */
     return pretrazi(koren->desno, dan, mesec);
   /* Funkcija koja pronalazi najmanji datum u stablu */
103
   Cvor *pronadji_najmanji_datum(Cvor * koren)
     /* Stablo je prazno, obustavlja se pretraga */
     if (koren == NULL)
       return NULL:
     /* Ako ne postoji leva grana korena, zbog uredjenja stabla koren
        sadrzi najmanji datum */
     if (koren->levo == NULL)
      return koren;
113
     else
       /* Inace, trazimo manji datum u levom podstablu */
       return pronadji_najmanji_datum(koren->levo);
  }
   /* Funkcija koja za dati dan i mesec odredjuje nisku formata DD.MM.
   void datum_u_nisku(int dan, int mesec, char datum[])
     if (dan < 10) {
       datum[0] = '0';
       datum[1] = dan + '0';
     } else {
       datum[0] = dan / 10 + '0';
       datum[1] = dan % 10 + '0';
     datum[2] = '.';
129
     if (mesec < 10) {
       datum[3] = '0';
       datum[4] = mesec + '0';
     } else {
       datum[3] = mesec / 10 + '0';
135
       datum[4] = mesec % 10 + '0';
     datum[5] = '.';
     datum[6] = '\0';
139
141
   /* Funkcija koja oslobadja memoriju zauzetu stablom */
  void oslobodi_stablo(Cvor ** adresa_korena)
143
     /* Stablo je prazno */
145
     if (*adresa_korena == NULL)
      return;
147
     /* Oslobadja se memorija zauzeta levim podstablom (ako postoji) */
```

```
if ((*adresa korena)->levo)
       oslobodi_stablo(&(*adresa_korena)->levo);
     /* Oslobadja se memorija zauzeta desnim podstablom (ako postoji) */
153
     if ((*adresa korena)->desno)
       oslobodi_stablo(&(*adresa_korena)->desno);
     /* Oslobadja se memorija zauzeta korenom */
     free(*adresa_korena);
159
     /* Proglasava se stablo praznim */
     *adresa_korena = NULL;
161
   int main(int argc, char **argv)
165 {
    FILE *in;
    Cvor *koren:
167
     Cvor *slavljenik;
     char ime[MAX_NISKA], prezime[MAX_NISKA];
     int dan, mesec;
    char datum[7];
     /* Provera da li je zadato ime ulazne datoteke */
     if (argc < 2) {
       /* Ako nije, ispisuje se poruka i prekida se sa izvrsavanjem
          programa */
       fprintf(stderr, "Greska: Nedostaje ime ulazne datoteke!\n");
       exit(EXIT_FAILURE);
179
     /* Inace, priprema se datoteka za citanje */
     in = fopen(argv[1], "r");
     if (in == NULL) {
183
       fprintf(stderr, "Greska: Neuspesno otvaranje datoteke %s.\n",
               argv[1]);
185
       exit(EXIT_FAILURE);
187
     /* I stablo se popunjava podacima uz proveru uspesnosti dodavanja
189
     koren = NULL:
     while (fscanf
            (in, "%s %s %d.%d.", ime, prezime, &dan, &mesec) != EOF)
       if (dodaj_u_stablo(&koren, ime, prezime, dan, mesec) == 1) {
         fprintf(stderr,
                 "Greska: Neuspelo dodavanje podataka za %s %s.\n", ime,
195
                 prezime);
         oslobodi_stablo(&koren);
197
         exit(EXIT_FAILURE);
199
```

```
/* Datoteka se zatvara */
201
     fclose(in);
203
     /* Omogucuje se pretraga podataka */
     while (1) {
205
       /* Ucitava se novi datum */
207
       printf("Unesite datum: "):
       if (scanf("%d.%d.", &dan, &mesec) == EOF)
209
         break:
       /* Pretrazuje se stablo */
       slavljenik = pretrazi(koren, dan, mesec);
213
       /* Ispisuju se pronadjeni podaci */
       /* Ako slavljenik nije pronadjen, to moze znaci da: */
       /* 1. drvo je prazno */
       if (slavljenik == NULL && koren == NULL) {
         printf("Nema podataka o ovom ni o sledecem rodjendanu.\n");
         continue;
       /* 2. posle datuma koji je unesen, nema podataka u stablu - u
          ovom slucaju se pretraga vrsi pocevsi od naredne godine i
          ispisuje se najmanji datum */
       if (slavljenik == NULL) {
         slavljenik = pronadji_najmanji_datum(koren);
         datum_u_nisku(slavljenik->dan, slavljenik->mesec, datum);
         printf("Slavljenik: %s %s %s\n", slavljenik->ime,
                slavljenik->prezime, datum);
231
         continue;
       /* Ako je slavljenik pronadjen, razlikuju se slucajevi: */
       /* 1. Pronadjeni su tacni podaci */
       if (slavljenik->dan == dan && slavljenik->mesec == mesec) {
         printf("Slavljenik: %s %s\n", slavljenik->ime,
                slavljenik->prezime);
239
         continue;
241
       /* 2. Pronadjeni su podaci o prvom sledecem rodjendanu */
       datum_u_nisku(slavljenik->dan, slavljenik->mesec, datum);
       printf("Slavljenik: %s %s %s\n", slavljenik->ime,
              slavljenik->prezime, datum);
245
247
     /* Oslobadja se memorija zauzeta stablom */
     oslobodi_stablo(&koren);
     exit(EXIT_SUCCESS);
```

Napomena: Rešenje koristi biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

```
#include <stdio.h>
2 #include <stdlib.h>
4 /* Ukljucuje se biblioteka za rad sa stablima */
  #include "stabla.h"
  /* Funkcija koja proverava da li su dva stabla koja sadrze cele
    brojeve identicna. Povratna vrednost funkcije je 1 ako jesu,
     odnosno 0 ako nisu */
int identitet(Cvor * koren1, Cvor * koren2)
    /* Ako su oba stabla prazna, jednaka su */
12
    if (koren1 == NULL && koren2 == NULL)
     return 1;
14
    /* Ako je jedno stablo prazno, a drugo nije, stabla nisu jednaka */
    if (koren1 == NULL || koren2 == NULL)
      return 0;
18
    /* Ako su oba stabla neprazna i u korenu se nalaze razlicite
       vrednosti, moze se zakljuciti da se razlikuju */
    if (koren1->broj != koren2->broj)
      return 0;
    /* Inace, proverava se da li vazi jednakost i levih i desnih
      podstabala */
26
    return (identitet(koren1->levo, koren2->levo)
            && identitet(koren1->desno, koren2->desno));
28
  }
30
  int main()
32 1
   int broj;
34
  Cvor *koren1, *koren2;
    /* Ucitavaju se elementi prvog stabla */
    koren1 = NULL;
    printf("Prvo stablo: ");
38
    scanf("%d", &broj);
    while (broj != 0) {
40
      if (dodaj_u_stablo(&koren1, broj) == 1) {
        fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n",
42
                broj);
        oslobodi_stablo(&koren1);
44
        exit(EXIT_FAILURE);
46
      scanf("%d", &broj);
```

```
48
    /* Ucitavaju se elementi drugog stabla */
    koren2 = NULL;
    printf("Drugo stablo: ");
    scanf("%d", &broj);
    while (broj != 0) {
54
      if (dodaj_u_stablo(&koren2, broj) == 1) {
        fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n",
                broj);
        oslobodi_stablo(&koren2);
58
        exit(EXIT_FAILURE);
      scanf("%d", &broj);
    /* Poziva se funkcija koja ispituje identitet stabala i ispisuje
64
       se njen rezultat */
    if (identitet(koren1, koren2))
      printf("Stabla jesu identicna.\n");
    else
68
      printf("Stabla nisu identicna.\n");
    /* Oslobadja se memorija zauzeta stablima */
    oslobodi_stablo(&koren1);
    oslobodi_stablo(&koren2);
74
    exit(EXIT_SUCCESS);
  }
76
```

Napomena: Rešenje koristi biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

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

/* Ukljucuje se biblioteka za rad sa stablima */
#include "stabla.h"

/* Funkcija kreira novo stablo identicno stablu koje je dato
korenom. Povratna vrednost funkcije je 0 ukoliko je kopiranje
uspesno, odnosno 1 ukoliko je doslo do greske */
int kopiraj_stablo(Cvor * koren, Cvor ** duplikat)

{
    /* Izlaz iz rekurzije */
    if (koren == NULL) {
        *duplikat = NULL;
        return 0;
}
```

```
/* Duplira se koren stabla i postavlja da bude koren novog stabla
18
    *duplikat = napravi_cvor(koren->broj);
    if (*duplikat == NULL) {
20
      return 1;
    /* Rekurzivno se duplirju levo i desno podstablo i njihove adrese
24
       se cuvaju redom u pokazivacima na levo i desno podstablo korena
       duplikata */
26
    int kopija_levo = kopiraj_stablo(koren->levo, &(*duplikat)->levo);
    int kopija_desno =
28
        kopiraj_stablo(koren->desno, &(*duplikat)->desno);
30
    /* Ako je uspesno duplirano i levo i desno podstablo */
    if (kopija_levo == 0 && kopija_desno == 0)
      /* Uspesno je duplirano i celo stablo */
     return 0;
34
    /* Inace, prijavljuje se da je doslo do greske */
    return 1;
36
38
  /* Funkcija izracunava uniju dva skupa predstavljena stablima -
     rezultujuci skup tj. stablo se dobija modifikacijom prvog stabla.
40
     Povratna vrednost funkcije je 0 ukoliko je kreiranje unije
    uspesno, odnosno 1 ukoliko je doslo do greske */
42
  int kreiraj_uniju(Cvor ** adresa_korena1, Cvor * koren2)
44
    /* Ako drugo stablo nije prazno */
    if (koren2 != NULL) {
46
      /* 1. Dodaje se njegov koren u prvo stablo */
      if (dodaj_u_stablo(adresa_korena1, koren2->broj) == 1) {
48
        return 1;
      /* 2. Rekurzivno se racuna unija levog i desnog podstabla drugog
52
         stabla sa prvim stablom */
      int unija_levo = kreiraj_uniju(adresa_korena1, koren2->levo);
      int unija_desno = kreiraj_uniju(adresa_korena1, koren2->desno);
56
      /* Ako je unija podstabala uspesno kreirana */
      if (unija_levo == 0 && unija_desno == 0)
        /* Uspesno je kreirana i unija stabala */
        return 0:
      /* U suprotnom se prijavljuje da je doslo do greske */
      return 1;
    }
64
    /* Ako je drugo stablo prazno, nista se ne preduzima */
    return 0:
```

```
68 }
   /* Funkcija izracunava presek dva skupa predstavljana stablima -
      rezultujuci skup tj. stablo se dobija modifikacijom prvog stabla.
      Povratna vrednost funkcije je 0 ukoliko je kreiranje preseka
      uspesno, odnosno 1 ukoliko je doslo do greske */
  int kreiraj_presek(Cvor ** adresa_korena1, Cvor * koren2)
74
     /* Ako je prvo stablo prazno, tada je i rezultat prazno stablo */
     if (*adresa_korena1 == NULL)
      return 0:
78
     /* Inace... Kreira se presek levog i desnog podstabla sa drugim
80
        stablom, tj. iz levog i desnog podstabla prvog stabla brisu se
        svi oni elementi koji ne postoje u drugom stablu */
82
     int presek_levo = kreiraj_presek(&(*adresa_korena1)->levo, koren2);
     int presek_desno =
84
         kreiraj_presek(&(*adresa_korena1)->desno, koren2);
     if (presek_levo == 0 && presek_desno == 0) {
86
       /* Ako se koren prvog stabla ne nalazi u drugom stablu tada se
          on uklanja iz prvog stabla */
88
       if (pretrazi_stablo(koren2, (*adresa_korena1)->broj) == NULL)
         obrisi_element(adresa_korena1, (*adresa_korena1)->broj);
90
       /* Presek stabala je uspesno kreiran */
92
       return 0;
94
     /* Inece, prijavljuje se da je doslo do greske */
     return 1;
96
98
   /* Funkcija izracunava razliku dva skupa predstavljana stablima -
      rezultujuci skup tj. stablo se dobija modifikacijom prvog stabla.
100
      Povratna vrednost funkcije je 0 ukoliko je kreiranje razlike
      uspesno, odnosno 1 ukoliko je doslo do greske */
   int kreiraj_razliku(Cvor ** adresa_korena1, Cvor * koren2)
104
     /* Ako je prvo stablo prazno, tada je i rezultat prazno stablo */
     if (*adresa_korena1 == NULL)
106
       return 0:
108
     /* Inace... */
     /* Kreira se razlika levog i desnog podstabla sa drugim stablom,
        tj. iz levog i desnog podstabla prvog stabla se brisu svi oni
        elementi koji postoje i u drugom stablu */
     int razlika_levo =
         kreiraj_razliku(&(*adresa_korena1)->levo, koren2);
114
     int razlika_desno =
         kreiraj_razliku(&(*adresa_korena1)->desno, koren2);
     if (razlika_levo == 0 && razlika_desno == 0) {
      /* Ako se koren prvog stabla nalazi i u drugom stablu tada se on
118
          uklanja se iz prvog stabla */
```

```
120
       if (pretrazi_stablo(koren2, (*adresa_korena1)->broj) != NULL)
         obrisi_element(adresa_korena1, (*adresa_korena1)->broj);
       /* Razlika stabala je uspesno kreirana */
       return 0:
124
     /* Inace, prijavljuje se da je doslo do greske */
     return 1;
128
130
   int main()
132 | {
     Cvor *skup1;
     Cvor *skup2;
134
     Cvor *pomocni_skup = NULL;
     int n;
136
     /* Ucitavaju se elementi prvog skupa */
138
     skup1 = NULL;
     printf("Prvi skup: ");
140
     while (scanf("%d", &n) != EOF) {
       if (dodaj_u_stablo(&skup1, n) == 1) {
         fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n", n);
         oslobodi_stablo(&skup1);
144
         exit(EXIT_FAILURE);
       }
146
     }
148
     /* Ucitavaju se elementi drugog skupa */
     skup2 = NULL;
     printf("Drugi skup: ");
     while (scanf("%d", &n) != EOF) {
       if (dodaj_u_stablo(&skup2, n) == 1) {
         fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n", n);
         oslobodi_stablo(&skup2);
         exit(EXIT_FAILURE);
       }
     }
158
     /* Kreira se unija skupova: prvo se napravi kopija prvog skupa
        kako bi se polazni skup mogao iskoristiti i za preostale
        operacije */
162
     if (kopiraj_stablo(skup1, &pomocni_skup) == 1) {
       oslobodi_stablo(&skup1);
164
       oslobodi_stablo(&pomocni_skup);
       exit(EXIT_FAILURE);
     }
     if (kreiraj_uniju(&pomocni_skup, skup2) == 1) {
       oslobodi_stablo(&pomocni_skup);
       oslobodi_stablo(&skup2);
170
       exit(EXIT_FAILURE);
```

```
printf("Unija: ");
     ispisi_stablo_infiksno(pomocni_skup);
174
     putchar('\n');
     /* Oslobadja se memorija zauzeta pomocnim skupom sa rezultatom
        operacije */
178
     oslobodi_stablo(&pomocni_skup);
180
     /* Kreira se presek skupova: prvo se napravi kopija prvog skupa
        kako bi se polazni skup mogao iskoristiti i za preostale
182
        operacije */
     if (kopiraj_stablo(skup1, &pomocni_skup) == 1) {
184
       oslobodi_stablo(&skup1);
       oslobodi_stablo(&pomocni_skup);
186
       exit(EXIT_FAILURE);
188
     if (kreiraj_presek(&pomocni_skup, skup2) == 1) {
       oslobodi_stablo(&pomocni_skup);
190
       oslobodi_stablo(&skup2);
       exit(EXIT_FAILURE);
     printf("Presek: ");
194
     ispisi_stablo_infiksno(pomocni_skup);
     putchar('\n');
196
     /* Oslobadja se memorija zauzeta pomocnim skupom sa rezultatom
198
        operacije */
     oslobodi_stablo(&pomocni_skup);
200
202
     /* Kreira se razlika skupova: prvo se napravi kopija prvog skupa
        kako bi se polazni skup mogao iskoristiti i za preostale
204
        operacije */
     if (kopiraj_stablo(skup1, &pomocni_skup) == 1) {
206
       oslobodi_stablo(&skup1);
       oslobodi_stablo(&pomocni_skup);
       exit(EXIT_FAILURE);
208
     if (kreiraj_razliku(&pomocni_skup, skup2) == 1) {
210
       oslobodi_stablo(&pomocni_skup);
       oslobodi_stablo(&skup2);
212
       exit(EXIT_FAILURE);
214
     printf("Razlika: ");
     ispisi_stablo_infiksno(pomocni_skup);
     putchar('\n');
218
     /* Oslobadja se memorija zauzeta pomocnim skupom sa rezultatom
220
        operacije */
     oslobodi_stablo(&pomocni_skup);
     /* Oslobadja se memorija zauzeta polaznim skupovima */
```

```
#include <stdio.h>
  #include <stdlib.h>
  /* Ukljucuje se biblioteka za rad sa stablima */
 #include "stabla.h"
7 #define MAX 50
  /* Funkcija koja obilazi stablo sa leva na desno i smesta vrednosti
     cvorova u niz. Povratna vrednost funkcije je broj vrednosti koje
    su smestene u niz */
  int kreiraj_niz(Cvor * koren, int a[])
13 {
    /* Stablo je prazno - u niz je smesteno 0 elemenata */
   if (koren == NULL)
17
      return 0;
19
    /* Dodaju se u niz elementi iz levog podstabla */
    r = kreiraj_niz(koren->levo, a);
21
    /* Tekuca vrednost promenljive r je broj elemenata koji su upisani
       u niz i na osnovu nje se moze odrediti indeks novog elementa */
    /* Smesta se vrednost iz korena */
    a[r] = koren->broj;
    /* Dodaju se elementi iz desnog podstabla */
    s = kreiraj_niz(koren->desno, a + r + 1);
    /* Racuna se indeks na koji treba smestiti naredni element */
    return r + s + 1;
  }
35
  /* Funkcija sortira niz tako sto najpre elemente niza smesti u
     stablo, a zatim kreira novi niz prolazeci kroz stablo sa leva na
     desno. Povratna vrednost funkcije je 0 ukoliko je niz uspesno
     kreiran i sortiran, a 1 ukoliko je doslo do greske.
39
```

```
Ovaj nacin sortiranja je primer sortiranja koje nije "u mestu"
     kao sto je to slucaj sa ostalim opisanim algoritmima sortiranja
     jer se sortiranje vrsi u pomocnoj dinamickoj strukturi, a ne
43
     razmenom elemenata niza. */
  int sortiraj(int a[], int n)
45
    int i;
47
    Cvor *koren:
49
    /* Kreira se stablo smestanjem elemenata iz niza u stablo */
    koren = NULL:
5.1
    for (i = 0; i < n; i++) {
      if (dodaj_u_stablo(&koren, a[i]) == 1) {
        oslobodi_stablo(&koren);
        return 1;
      }
    /* Infiksnim obilaskom stabla elementi iz stabla se prepisuju u
       niz a */
    kreiraj_niz(koren, a);
61
    /* Stablo vise nije potrebno pa se oslobadja zauzeta memorija */
    oslobodi_stablo(&koren);
63
    /* Vraca se indikator uspesnog sortiranja */
65
    return 0;
  }
67
  int main()
69
    int a[MAX];
    int n, i;
73
    /* Ucitavaju se dimenzija i elementi niza */
    printf("n: ");
    scanf("%d", &n);
    if (n < 0 | | n > MAX) {
      printf("Greska: Pogresna dimenzija niza!\n");
      exit(EXIT_FAILURE);
79
81
    printf("a: ");
    for (i = 0; i < n; i++)
83
      scanf("%d", &a[i]);
85
    /* Poziva se funkcija za sortiranje */
    if (sortiraj(a, n) == 0) {
87
      /* Ako je niz uspesno sortiran, ispisuje se rezultujuci niz */
      for (i = 0; i < n; i++)
89
        printf("%d ", a[i]);
      printf("\n");
    } else {
```

```
/* Inace, obavestava se korisnik da je doslo do greske */
printf("Greska: Problem prilikom sortiranja niza!\n");

/* Problem prilikom sortiranja niza!\n");

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
2 #include <stdlib.h>
4 /* Ukljucuje se biblioteka za rad sa stablima */
  #include "stabla.h"
  /* a) Funkcija koja izracunava broj cvorova stabla */
8 int broj_cvorova(Cvor * koren)
    /* Ako je stablo prazno, broj cvorova je nula */
   if (koren == NULL)
     return 0;
12
14
    /* U suprotnom je broj cvorova stabla jednak zbiru broja cvorova u
       levom podstablu i broja cvorova u desnom podstablu - 1 se dodaje
      zato sto treba racunati i koren */
    return broj_cvorova(koren->levo) + broj_cvorova(koren->desno) + 1;
18 }
20 /* b) Funkcija koja izracunava broj listova stabla */
  int broj_listova(Cvor * koren)
    /* Ako je stablo prazno, broj listova je nula */
24
   if (koren == NULL)
      return 0;
26
    /* Proverava se da li je tekuci cvor list */
    if (koren->levo == NULL && koren->desno == NULL)
      /* Ako jeste vraca se 1 - to ce kasnije zbog rekurzivnih poziva
         uvecati broj listova za 1 */
30
     return 1;
    /* U suprotnom se prebrojavaju listovi koje se nalaze u
       podstablima */
34
    return broj_listova(koren->levo) + broj_listova(koren->desno);
36 }
38 /* c) Funkcija koja stampa pozitivne vrednosti listova stabla */
 | void pozitivni_listovi(Cvor * koren)
```

```
40 {
    /* Slucaj kada je stablo prazno */
    if (koren == NULL)
42
      return;
44
    /* Ako je cvor list i sadrzi pozitivnu vrednost */
    if (koren->levo == NULL && koren->desno == NULL && koren->broj > 0)
46
      /* Stampa se */
      printf("%d ", koren->broj);
48
    /* Nastavlja se sa stampanjem pozitivnih listova u podstablima */
    pozitivni listovi(koren->levo);
    pozitivni_listovi(koren->desno);
54
  /* d) Funkcija koja izracunava zbir cvorova stabla */
  int zbir_svih_cvorova(Cvor * koren)
56
    /* Ako je stablo prazno, zbir cvorova je 0 */
58
    if (koren == NULL)
      return 0;
60
    /* Inace, zbir cvorova stabla izracunava se kao zbir korena i svih
       elemenata u podstablima */
    return koren->broj + zbir_svih_cvorova(koren->levo) +
64
        zbir_svih_cvorova(koren->desno);
  }
66
  /* e) Funkcija koja izracunava najveci element stabla */
  Cvor *najveci_element(Cvor * koren)
70
    /* Ako je stablo prazno, obustavlja se pretraga */
    if (koren == NULL)
      return NULL;
74
    /* Zbog prirode pretrazivackog stabla, vrednosti vece od korena se
       nalaze u desnom podstablu */
76
    /* Ako desnog podstabla nema */
    if (koren->desno == NULL)
      /* Najveca vrednost je koren */
80
      return koren;
82
    /* Inace, najveca vrednost se trazi desno */
    return najveci_element(koren->desno);
84
86
  /* f) Funkcija koja izracunava dubinu stabla */
88 int dubina_stabla(Cvor * koren)
    /* Dubina praznog stabla je 0 */
90
    if (koren == NULL)
```

```
92
       return 0;
     /* Izracunava se dubina levog podstabla */
94
     int dubina_levo = dubina_stabla(koren->levo);
96
     /* Izracunava se dubina desnog podstabla */
     int dubina_desno = dubina_stabla(koren->desno);
98
     /* Dubina stabla odgovara vecoj od dubina podstabala - 1 se dodaje
        jer se racuna i koren */
     return dubina_levo >
         dubina_desno ? dubina_levo + 1 : dubina_desno + 1;
104 }
106 /* g) Funkcija koja izracunava broj cvorova na i-tom nivou stabla */
   int broj_cvorova_na_itom_nivou(Cvor * koren, int i)
108 {
     /* Ideja je spustanje kroz stablo sve dok se ne stigne do trazenog
        nivoa */
     /* Ako nema vise cvorova, nema spustanja niz stablo */
     if (koren == NULL)
      return 0;
114
     /* Ako se stiglo do trazenog nivoa, vraca se 1 - to ce kasnije
116
        zbog rekurzivnih poziva uvecati broj cvorova za 1 */
     if (i == 0)
118
      return 1;
     /* Inace, spusta se jedan nivo nize i u levom i u desnom postablu
     return broj_cvorova_na_itom_nivou(koren->levo, i - 1)
         + broj_cvorova_na_itom_nivou(koren->desno, i - 1);
124 }
126 /* h) Funkcija koja ispisuje sve elemente na i-tom nivou */
   void ispis_nivo(Cvor * koren, int i)
128 {
     /* Nema vise cvorova, nema spustanja kroz stablo */
     if (koren == NULL)
130
      return;
     /* Ako se stiglo do trazenog nivoa - ispisuje se vrednost */
     if (i == 0) {
134
       printf("%d ", koren->broj);
       return;
136
     }
     /* Inace, spustanje se nastavlja za jedan nivo nize i u levom i u
138
        desnom podstablu */
     ispis_nivo(koren->levo, i - 1);
140
     ispis_nivo(koren->desno, i - 1);
142 }
```

```
/* i) Funkcija koja izracunava maksimalnu vrednost na i-tom nivou
      stabla */
  Cvor *najveci_element_na_itom_nivou(Cvor * koren, int i)
146
     /* Ako je stablo prazno, obustavlja se pretraga */
148
     if (koren == NULL)
       return NULL:
     /* Ako se stiglo do trazenog nivoa, takodje se prekida pretraga */
     if (i == 0)
       return koren;
154
     /* Pronalazi se maksimum na i-tom nivou levog podstabla */
     Cvor *a = najveci_element_na_itom_nivou(koren->levo, i - 1);
158
     /* Pronalazi se maksimum na i-tom nivou desnog podstabla */
     Cvor *b = najveci_element_na_itom_nivou(koren->desno, i - 1);
160
     /* Trazi se i vraca maksimum izracunatih vrednosti */
162
     if (a == NULL && b == NULL)
      return NULL;
164
     if (a == NULL)
      return b;
     if (b == NULL)
      return a;
168
     /* Ako su obe vrednosti razlicite od NULL, veca od vrednosti se
        nalazi u b cvoru jer je stablo pretrazivacko */
     return b;
  }
   /* j) Funkcija koja izracunava zbir cvorova na i-tom nivou */
   int zbir_cvorova_na_itom_nivou(Cvor * koren, int i)
176
     /* Ako je stablo prazno, zbir je nula */
     if (koren == NULL)
178
       return 0;
180
     /* Ako se stiglo do trazenog nivoa, vraca se vrednost */
     if (i == 0)
182
       return koren->broj;
184
     /* Inace, spustanje se nastavlja za jedan nivo nize i traze se
        sume iz levog i desnog podstabla */
     return zbir_cvorova_na_itom_nivou(koren->levo, i - 1)
         + zbir_cvorova_na_itom_nivou(koren->desno, i - 1);
188
190
  /* k) Funkcija koja izracunava zbir svih vrednosti u stablu koje su
      manje ili jednake od date vrednosti x */
int zbir_manjih_od_x(Cvor * koren, int x)
```

```
/* Ako je stablo prazno, zbir je nula */
     if (koren == NULL)
       return 0;
     /* Ako je vrednost u korenu manja od trazene vrednosti, zbog
200
        prirode pretrazivackog stabla treba obici i levo i desno
        podstablo */
202
     if (koren->broj <= x)
      return koren->broj + zbir_manjih_od_x(koren->levo, x) +
204
           zbir_manjih_od_x(koren->desno, x);
206
     /* Inace, racuna se samo suma vrednosti iz levog podstabla jer
        medju njima jedino moze biti onih koje zadovoljavaju uslov */
208
     return zbir_manjih_od_x(koren->levo, x);
210 }
212 int main(int argc, char **argv)
     /* Analiza argumenata komandne linije */
214
     if (argc != 3) {
      fprintf(stderr,
               "Greska: Program se poziva sa: ./a.out nivo
       broj_za_pretragu\n");
       exit(EXIT_FAILURE);
218
     int i = atoi(argv[1]);
220
     int x = atoi(argv[2]);
     /* Kreira se stablo uz proveru uspesnosti dodavanja novih
        vrednosti */
224
     Cvor *koren = NULL;
     int broj;
     while (scanf("%d", &broj) != EOF) {
       if (dodaj_u_stablo(&koren, broj) == 1) {
228
         fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n",
230
                 broj);
         oslobodi_stablo(&koren);
         exit(EXIT_FAILURE);
       }
     }
234
     /* Ispisuju se rezultati rada funkcija */
     printf("Broj cvorova: %d\n", broj_cvorova(koren));
     printf("Broj listova: %d\n", broj_listova(koren));
238
     printf("Pozitivni listovi: ");
     pozitivni_listovi(koren);
240
     printf("\n");
     printf("Zbir cvorova: %d\n", zbir_svih_cvorova(koren));
     if (najveci_element(koren) == NULL)
       printf("Najveci element: ne postoji\n");
244
     else
```

```
printf("Najveci element: %d\n", najveci_element(koren)->broj);
246
     printf("Dubina stabla: %d\n", dubina_stabla(koren));
248
     printf("Broj cvorova na %d. nivou: %d\n", i,
250
            broj_cvorova_na_itom_nivou(koren, i));
     printf("Elementi na %d. nivou: ", i);
252
     ispis_nivo(koren, i);
     printf("\n");
     if (najveci_element_na_itom_nivou(koren, i) == NULL)
       printf("Nema elemenata na %d. nivou!\n", i);
     else
       printf("Maksimalni element na %d. nivou: %d\n", i,
258
              najveci_element_na_itom_nivou(koren, i)->broj);
260
     printf("Zbir elemenata na %d. nivou: %d\n", i,
            zbir_cvorova_na_itom_nivou(koren, i));
262
     printf("Zbir elemenata manjih ili jednakih od %d: %d\n", x,
            zbir_manjih_od_x(koren, x));
264
     /* Oslobadja se memorija zauzeta stablom */
266
     oslobodi_stablo(&koren);
268
     exit(EXIT_SUCCESS);
   }
```

```
#include <stdio.h>
  #include <stdlib.h>
  /* Ukljucuje se biblioteka za rad sa stablima */
  #include "stabla.h"
  /* Funkcija koja izracunava dubinu stabla */
  int dubina_stabla(Cvor * koren)
    /* Dubina praznog stabla je 0 */
    if (koren == NULL)
      return 0:
12
    /* Izracunava se dubina levog podstabla */
    int dubina_levo = dubina_stabla(koren->levo);
16
    /* Izracunava se dubina desnog podstabla */
    int dubina_desno = dubina_stabla(koren->desno);
18
    /* Dubina stabla odgovara vecoj od dubina podstabala - 1 se dodaje
```

```
jer se racuna i koren */
    return dubina levo >
        dubina_desno ? dubina_levo + 1 : dubina_desno + 1;
24 }
26 /* Funkcija koja ispisuje sve elemente na i-tom nivou */
  void ispisi_nivo(Cvor * koren, int i)
28 | {
    /* Nema vise cvorova, nema spustanja niz stablo */
    if (koren == NULL)
30
      return;
    /* Ako se stiglo do trazenog nivoa - ispisuje se vrednost */
    if (i == 0) {
      printf("%d ", koren->broj);
      return;
36
    /* Inace, vrsi se spustanje za jedan nivo nize i u levom i u
38
       desnom podstablu */
    ispisi_nivo(koren->levo, i - 1);
40
    ispisi_nivo(koren->desno, i - 1);
42 }
44 /* Funkcija koja ispisuje stablo po nivoima */
  void ispisi_stablo_po_nivoima(Cvor * koren)
46 {
    int i;
48
    /* Prvo se izracunava dubina stabla */
    int dubina;
    dubina = dubina_stabla(koren);
52
    /* Ispisuje se nivo po nivo stabla */
    for (i = 0; i < dubina; i++) {
54
      printf("%d. nivo: ", i);
      ispisi_nivo(koren, i);
      printf("\n");
58
  int main(int argc, char **argv)
62 | {
    Cvor *koren;
    int broj;
64
    /* Citaju se vrednosti sa ulaza i dodaju se u stablo uz proveru
       uspesnosti dodavanja */
    koren = NULL;
68
    while (scanf("%d", &broj) != EOF) {
      if (dodaj_u_stablo(&koren, broj) == 1) {
        fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n",
72
                 broj);
```

```
oslobodi_stablo(&koren);
exit(EXIT_FAILURE);
}

/* Ispisuje se stablo po nivoima */
ispisi_stablo_po_nivoima(koren);

/* Oslobadja se memorija zauzeta stablom */
oslobodi_stablo(&koren);

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
  #include <stdlib.h>
  /* Ukljucuje se biblioteka za rad sa stablima */
5 #include "stabla.h"
7 /* Funkcija koja izracunava dubinu stabla */
  int dubina_stabla(Cvor * koren)
    /* Dubina praznog stabla je 0 */
    if (koren == NULL)
      return 0;
13
    /* Izracunava se dubina levog podstabla */
    int dubina_levo = dubina_stabla(koren->levo);
15
    /* Izracunava se dubina desnog podstabla */
    int dubina_desno = dubina_stabla(koren->desno);
19
    /* Dubina stabla odgovara vecoj od dubina podstabala - 1 se dodaje
       jer se racuna i koren */
    return dubina_levo >
        dubina_desno ? dubina_levo + 1 : dubina_desno + 1;
25
  /* Funkcija koja racuna broj cvorova koji ispunjavaju uslov za AVL
     stablo */
  int avl(Cvor * koren)
    int dubina_levo, dubina_desno;
    /* Ako je stablo prazno, zaustavlja se brojanje */
```

```
if (koren == NULL) {
33
      return 0;
    /* Izracunava se dubina levog podstabla korena */
37
    dubina_levo = dubina_stabla(koren->levo);
39
    /* Izracunava se dubina desnog podstabla korena */
    dubina_desno = dubina_stabla(koren->desno);
41
    /* Ako je uslov za AVL stablo ispunjen */
43
    if (abs(dubina_desno - dubina_levo) <= 1) {</pre>
      /* Racuna se broj AVL cvorova u levom i desnom podstablu i
45
         uvecava za jedan iz razloga sto koren ispunjava uslov */
      return 1 + avl(koren->levo) + avl(koren->desno);
47
    } else {
      /* Inace, racuna se samo broj AVL cvorova u podstablima */
49
      return avl(koren->levo) + avl(koren->desno);
  }
53
  int main(int argc, char **argv)
55 {
    Cvor *koren;
    int broj;
    /* Ucitavaju se vrednosti sa ulaza i dodaju u stablo uz proveru
59
       uspesnosti dodavanja */
    koren = NULL;
    while (scanf("%d", &broj) != EOF) {
      if (dodaj_u_stablo(&koren, broj) == 1) {
        fprintf(stderr, "Greska: Neuspelo dodavanje broja %d.\n",
                broj);
        oslobodi_stablo(&koren);
        exit(EXIT_FAILURE);
      }
    }
    /* Racuna se i ispisuje broj AVL cvorova */
    printf("%d\n", avl(koren));
73
    /* Oslobadja se memorija zauzeta stablom */
    oslobodi_stablo(&koren);
    exit(EXIT_SUCCESS);
```

```
#include <stdio.h>
  #include <stdlib.h>
  /* Ukljucuje se biblioteka za rad sa stablima */
  #include "stabla.h"
  /* Funkcija koja kreira stablo prema zadatoj slici. Povratna
     vrednost funkcije je 0 ako je stablo uspesno kreirano, odnosno 1
     ukoliko je doslo do greske */
  int kreiraj_hip(Cvor ** adresa_korena)
    /* Stablo se proglasava praznim */
12
    *adresa_korena = NULL;
14
    /* Dodaje se cvor po cvor uz proveru uspesnosti dodavanja */
    if (((*adresa_korena) = napravi_cvor(100)) == NULL)
      return 1;
    if (((*adresa_korena)->levo = napravi_cvor(19)) == NULL)
18
      return 1;
    if (((*adresa_korena)->levo->levo = napravi_cvor(17)) == NULL)
20
      return 1;
    if (((*adresa korena)->levo->levo->levo = napravi cvor(2)) == NULL)
      return 1;
    if (((*adresa_korena)->levo->levo->desno =
24
         napravi_cvor(7)) == NULL)
      return 1;
26
    if (((*adresa korena)->levo->desno = napravi cvor(3)) == NULL)
      return 1;
28
    if (((*adresa_korena)->desno = napravi_cvor(36)) == NULL)
      return 1;
30
    if (((*adresa_korena)->desno->levo = napravi_cvor(25)) == NULL)
      return 1:
    if (((*adresa_korena)->desno->desno = napravi_cvor(1)) == NULL)
      return 1:
34
    /* Vraca se indikator uspesnog kreiranja */
36
    return 0;
  }
38
  /* Funkcija proverava da li je zadato binarno stablo celih
40
     pozitivnih brojeva hip. Ideja koja ce biti implementirana u
     osnovi ima pronalazenje maksimalne vrednosti levog i maksimalne
42
     vrednosti desnog podstabla - ako je vrednost u korenu veca od
     izracunatih vrednosti, uoceni fragment stabla zadovoljava uslov
44
     za hip. Zato ce funkcija vracati maksimalne vrednosti iz uocenog
     podstabala ili vrednost -1 ukoliko se zakljuci da stablo nije
46
     hip. */
  int hip(Cvor * koren)
48
    int max_levo, max_desno;
```

```
/* Prazno sablo je hip - kao rezultat se vraca 0 kao najmanji
       pozitivan broj */
    if (koren == NULL) {
54
     return 0;
56
    /* Ukoliko je stablo list... */
    if (koren->levo == NULL && koren->desno == NULL) {
58
     /* Vraca se njegova vrednost */
     return koren->broj;
    }
    /* Inace... Proverava se svojstvo za levo podstablo */
    max_levo = hip(koren->levo);
64
    /* Proverava se svojstvo za desno podstablo */
    max_desno = hip(koren->desno);
68
    /* Ako levo ili desno podstablo uocenog cvora nije hip, onda nije
      ni celo stablo */
    if (max_levo == -1 || max_desno == -1) {
     return -1;
74
    /* U suprotonom proverava se da li svojstvo vazi za uoceni cvor */
    if (koren->broj > max_levo && koren->broj > max_desno) {
     /* Ako vazi, vraca se vrednost korena */
     return koren->broj;
78
    }
80
    /* U suprotnom se zakljucuje da stablo nije hip */
    return -1;
82
84
  int main(int argc, char **argv)
86 {
    Cvor *koren;
   int hip_indikator;
88
    /* Kreira se stablo prema zadatoj slici */
90
    if (kreiraj_hip(&koren) == 1) {
     fprintf(stderr, "Greska: Neuspesno kreiranje hipa.\n");
      oslobodi_stablo(&koren);
      exit(EXIT_FAILURE);
94
96
    /* Poziva se funkcija kojom se proverava da li je stablo hip */
    hip_indikator = hip(koren);
    /* Ispisuje se rezultat */
    if (hip_indikator == -1) {
     printf("Zadato stablo nije hip!\n");
    } else {
```

```
printf("Zadato stablo je hip!\n");
}

/* Oslobadja se memorija zauzeta stablom */
oslobodi_stablo(&koren);

exit(EXIT_SUCCESS);
}
```

# Dodatak A

# Ispitni rokovi

# A.1 Praktični deo ispita, jun 2015.

Zadatak A.1 Kao argument komandne linije zadaje se ime ulazne datoteke u kojoj se nalaze niske. U prvoj liniji datoteke nalazi se informacija o broju niski, a zatim u narednim linijama po jedna niska ne duža od 50 karaktera. Napisati program u kojem se dinamički alocira memorija za zadati niz niski, a zatim se na standardnom izlazu u redosledu suprotnom od redosleda čitanja ispisuju sve niske koje počinju velikim slovom. U slučaju pojave bilo kakve greške na standardnom izlazu za grešku ispisati vrednost -1 i prekinuti izvršavanje programa.

```
Test 1

| Pokretanje: ./a.out ulaz.txt

ULAZ.TXT
5
| Programiranje | Matematika | 12345 | dInAmiCnArEc | Ispit

| Izlaz: | Ispit | Matematika | Programiranje |
```

Zadatak A.2 Data je biblioteka za rad sa binarnim pretraživačkim stablima čiji čvorovi sadrže cele brojeve. Napisati funkciju int sumiraj\_n (Cvor \* koren, int n) koja izračunava zbir svih čvorova koji se nalaze na n-tom nivou stabla (koren se nalazi na nultom nivou, njegova deca na prvom nivou i tako redom). Ispravnost napisane funkcije testirati na osnovu zadate main funkcije i biblioteke za rad sa pretraživačkim stablima.

Napisati program koji sa standardnog ulaza učitava najpre prirodan broj n, a potom i brojeve sve do pojave nule koje smešta u stablo i ispisuje rezultat pozivanja funkcije sumiraj\_n za broj n i tako kreirano stablo. U slučaju greške na standardni izlaz za greške ispisati -1. Napomena: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima 4.14.

**Zadatak A.3** Sa standardnog ulaza učitava se broj vrsta i broj kolona celobrojne matrice A, a zatim i elementi matrice A. Napisati program koji će ispisati indeks kolone u kojoj se nalazi najviše negativnih elemenata. Ukoliko postoji više takvih kolona, ispisati indeks prve kolone. Može se pretpostaviti da je broj vrsta i broj kolona manji od 50. U slučaju greške ispisati vrednost -1 na standardni izlaz za greške.

```
Test 1
                              Test 2
                                                            Test 3
ULAZ:
                              ULAZ:
                                                            ULAZ:
 4 5
                               23
                                                              -2
 12345
                               0 0 -5
                                                            IZLAZ ZA GREŠKE:
  -1 2 -3 4 -5
                               12-4
  -5 -4 -3 -2 1
                              TZI.AZ:
 -1 0 0 0 0
IzLAz:
```

# A.2 Praktični deo ispita, jul 2015.

Zadatak A.4 Napisati program koji kao prvi arugment komandne linije prima ime dokumenta u kome treba prebrojati sva pojavljivanja tražene niske (bez preklapanja) koja se navodi kao drugi argument komandne linije (iskoristiti funkciju standardne biblioteke strstr). U slučaju bilo kakve greške ispisati –1 na standardni izlaz za greške. Pretpostaviti da linije datoteke neće biti duže od 127 karaktera.

Potpis funkcije strstr:

```
char *strstr(const char *haystack, const char *needle);
```

Funkcija traži prvo pojavljivanje podniske needle u nisci haystack, i vraća pokazivač na početak podniske, ili NULL ako podniska nije pronađena.

```
Test 1
                                                    Test 2
POKRETANJE: ./a.out ulaz.txt test
                                                  POKRETANJE: ./a.out
ULAZ.TXT
  Ovo je test primer.
  U niemu se rec test javlja
 vise puta. testtesttest
IZLAZ:
 Test 3
                                                    Test 4
POKRETANJE: ./a.out ulaz.txt foo
                                                  POKRETANJE: ./a.out ulaz.txt .
DATOTEKA ULAZ.TXT NE POSTOJI
                                                  DATOTEKA ULAZ.TXT JE PRAZNA
IZLAZ ZA GREŠKE:
```

Zadatak A.5 Na početku datoteke trouglovi.txt nalazi se broj trouglova čije su koordinate temena zapisane u nastavku datoteke. Napisati program koji učitva trouglove, i ispisuje ih na standardni izlaz sortirane po površini opadajuće (koristiti Heronov obrazac:  $P = \sqrt{s*(s-a)*(s-b)*(s-c)}$ , gde je s poluobim trougla). U slučaju bilo kakve greške ispisati -1 na standardni izlaz za greške. Ne praviti nikave pretpostavke o broju trouglova u datoteci, i proveriti da li je datoteka ispravno zadata.

```
Test 3

| DATOTEKA TROUGLOVI.TXT NE POSTOJI
| IZLAZ ZA GREŠKE: | TROUGLOVI.TXT | O | IZLAZ:
```

Zadatak A.6 Data je biblioteka za rad sa binarnim pretraživačkim stablima celih brojeva. Napisati funkciju

```
int prebroj_n(Cvor *koren, int n)
```

koja u datom stablu prebrojava čvorove na n-tom nivou, koji imaju tačno jednog potomka. Pretpostaviti da se koren nalazi na nivou 0. Ispravnost napisane funkcije testirati na osnovu zadate main funkcije i biblioteke za rad sa stablima. NAPOMENA: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

```
Test 1
                           Test 2
                                                      Test 3
                          ULAZ:
                                                    ULAZ:
  15361479
                            253610479
                                                       0425
IZLAZ:
                          IZLAZ:
                                                     IZLAZ:
                           Test 5
Test 4
ULAZ:
                          || ULAZ:
                            -1 4 5 1 7
  .3
IZLAZ:
                           Tzlaz:
 0
                            0
```

# A.3 Praktični deo ispita, septembar 2015.

**Zadatak A.7** Sa standardnog ulaza se učitavaju neoznačeni celi brojevi x i n. Na standardni izlaz ispisati neoznačen ceo broj koji se dobija od broja x kada se njegov binarni zapis rotira za n mesta udesno (na primer, ako je binarni zapis broja x jednak 0000000000000000000000001111, i ako je n=1 tada na standardni izlaz treba ispisati neočnačen broj čiji je binarni zapis jednak

10000000000000000000000000000111).

```
Test 1
                                Test 2
                                                               Test 3
ULAZ:
                               ULAZ:
                                                               ULAZ:
  6 1
                                 15 3
                                                                31 100
 IZLAZ:
                               IZLAZ:
                                                               IZLAZ:
  3
                                 3758096385
                                                                4026531841
                                Test 5
 Test 4
ULAZ:
                               ULAZ:
                                 0 5
 40
IZLAZ:
                                IZLAZ:
                                 0
  4
```

Zadatak A.8 Data je biblioteka za rad sa listama. Napisati funkciju int dopuni\_listu(Cvor \*\* adresa\_glave) koja samo čvorovima koji imaju sledbenika u jednostruko povezanoj listi realnih brojeva, dodaje između čvora i njegovog sledbenika nov čvor čija vrednost je aritmetička sredina njihovih vrednosti. Povratna vrednost funkcije treba da bude 1 ukoliko je došlo greške pri alokaciji memorije, inače 0. Ispravnost napisane funkcije testirati koristeći dostupnu biblioteku za rad sa listama i main funkciju koja najpre učitava elemente liste, poziva pomenutu funkciju i ispisuje sadržaj liste.

```
Test 1
ULAZ:
 12345
  1.00 1.50 2.00 2.50 3.00 3.50 4.00 4.50 5.00
 Test 2
                               Test 3
                                                             Test 4
ULAZ:
                               ULAZ:
                                                             ULAZ:
 12
                                prazna lista
                                                              13.3 15.8
IZLAZ:
                               IZLAZ:
                                                             IZLAZ:
  12.00
                                                               13.30 14.55
```

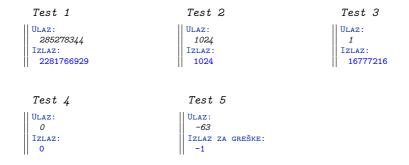
Zadatak A.9 Sa standardnog ulaza se učitava dimenzija n kvadratne celobrojne matrice A (n>0), a zatim i elementi matrice A. Napisati program koji proverava da li je data kvadratna matrica magični kvadrat (magični kvadrat je kvadratna matrica kod koje su sume brojeva u svim redovima i kolonama međusobno jednake). Ukoliko jeste, ispisati na standardnom izlazu sumu brojeva jedne vrste ili kolone te matrice, a ukoliko nije ispisati -". Broj vrsta i broj kolona matrice nije unapred poznat. U slučaju greške ispisati -1 na standardni izlaz za greške. Napomena: Koristiti biblioteku za rad sa celobrojnim matricama iz

zadatka 2.19.

```
Test 1
                              Test 2
                                                            Test 3
 ULAZ:
                              ULAZ:
                                                            ULAZ:
                                                             2
  1234
                               1 1 1
                                                             1 1
                                                             22
  2143
                               1 1 1
  3 4 2 1
                               1 1 1
                                                            IzLaz:
  4312
 IzLAz:
  10
 Test 4
                              Test 5
                                                            Test 6
ULAZ:
                                                          || ULAZ:
                              ULAZ:
 2
                                                             0
  12
                               5
                                                            Izlaz za greške:
  12
                              IzLAz:
                                                             -1
                               5
```

# A.4 Praktični deo ispita, januar 2016.

Zadatak A.10 Napisati funkciju unsigned int zamena (unsigned int x) koja u datom broju x menja mesta prvom i četvrtom bajtu. Prvi bajt je sačinjen od 8 bitova najmanje težine. Napisati program koji testira funkciju zamena za ceo broj unet sa standardnog ulaza. U slučaju da je uneti broj negativan, na standardni izlaz za greške program ispisuje -1, a inače ispisuje na standardni izlaz broj dobijen primenom funkcije zamena.



Zadatak A.11 Data je biblioteka za rad sa binarnim pretraživackim stablima celih brojeva. Napisati funkciju int najduzi\_put (Cvor \* koren) koja

za dato stablo izračunava dužinu najdužeg puta od korena do nekog lista. Ako je stablo prazno, povratna vrednost funkcije je -1. Ako stablo ima samo koren, dužina najdužeg puta je 0. Ispravnost napisane funkcije testirati na osnovu zadate main funkcije i biblioteke za rad sa stablima. Napomena: Koristiti biblioteku za rad sa binarnim pretraživačkim stablima celih brojeva iz zadatka 4.14.

```
Test 1
                                                    Test 2
|| ULAZ:
                                                   ULAZ:
  10 5 15 3 2 4 30 12 14 13
                                                     3
 IZLAZ:
                                                    IZLAZ:
  4
                                                     0
 Test 3
                                Test 4
                                                               Test 5
ULAZ:
                                ULAZ:
                                                               ULAZ:
                                                                 578
  5 6
                                 758
 IZLAZ:
                                IzLaz:
                                                               IzLAz:
                                                                 2
  1
                                 1
```

Zadatak A.12 Sa standardnog ulaza zadaje se ime datoteke u kojoj se nalazi matrica realnih brojeva jednostruke tačnosti i jedan realan broj. Napisati program koji iz datoteke učitava matricu realnih brojeva, a zatim pronalazi i na standardni izlaz ispisuje indeks vrste matrice u kojoj se uneti realan broj pojavljuje najmanje puta. Ako postoji više takvih vrsta, ispisati indeks prve vrste. U datoteci su prvo navedena dva cela broja koja predstavljaju dimenzije matrice, redom broj vrsta i broj kolona, a zatim i elementi matrice vrstu po vrstu. U slučaju greške ispisati –1 na standardni izlaz za greške. Pretpostaviti da ime datoteke neće biti duže od 30 karaktera. Napomena: U zadatku treba koristiti dinamičku alokaciju memorije.

```
Test 1
                                Test 2
                                                                Test 3
ULAZ:
                                ULAZ:
                                                                ULAZ:
 brojevi.txt 0
                                 in.txt 2
                                                                  brojevi.txt 12
BROJEVI.TXT
                                IN.TXT
                                                                DATOTEKA BROJEVI.TXT JE PRAZNA
 4 4
                                 3 3
 0 0 0 1.2
                                 2 0 2
                                                                IZLAZ ZA GREŠKE:
 1 0 0.3 0.3
                                 -1 2 -1
                                                                 -1
 0.5 0.5 0.9 -1
                                 2 5 3
 -2 0 0 0
IzLAz:
                                 1
 2
```

# A.5 Rešenja

```
#include <stdio.h>
2 #include <stdlib.h>
  #include <ctype.h>
4 #define MAKS 50
  /* Funkcija vrsi dinamicku alokaciju memorije potrebne n linija tj.
     n niski od kojih nijedna nije duza od MAKS karaktera. */
 char **alociranje_memorije(int n)
    char **linije = NULL;
    int i, j;
12
    /* Alocira se prostor za niz vrsti matrice */
    linije = (char **) malloc(n * sizeof(char *));
    /* U slucaju neuspesnog otvaranja ispisuje se -1 na stderr i
       program zavrsava. */
16
    if (linije == NULL)
      return NULL;
    /* Alocira se prostor za svaku vrstu matrice. Niska nije duza od
18
       MAKS karaktera, a 1 se dodaje zbog terminirajuce nule. */
    for (i = 0; i < n; i++) {
20
      linije[i] = malloc((MAKS + 1) * sizeof(char));
      /* Ako alokacija nije prosla uspesno, oslobadjaju se svi
         prethodno alocirani resursi, i povratna vrednost je NULL */
      if (linije[i] == NULL) {
        for (j = 0; j < i; j++) {
          free(linije[j]);
26
        free(linije);
        return NULL;
30
    return linije;
34
  /* Funkcija oslobadjaja dinamicki alociranu memoriju */
36 char **oslobadjanje_memorije(char **linije, int n)
  {
38
    /* Oslobadja se prostor rezervisan za svaku vrstu */
    for (i = 0; i < n; i++) {
40
      free(linije[i]);
42
    /* Oslobadja se memorija za niz pokazivaca na vrste */
    free(linije);
44
    /* Matrica postaje prazna, tj. nealocirana */
```

```
return NULL;
  }
48
50 int main(int argc, char *argv[])
    FILE *ulaz;
    char **linije;
    int i, n;
54
    /* Proverava argumenata komandne linije. */
56
    if (argc != 2) {
      fprintf(stderr, "-1\n");
58
      exit(EXIT_FAILURE);
    /* Otvara se datoteka cije ime je navedeno kao argument komandne
62
       linije neposredno nakon imena programa koji se poziva. U
       slucaju neuspesnog otvaranja ispisuje se -1 na stderr i program
64
       zavrsava izvrsavanje. */
    ulaz = fopen(argv[1], "r");
66
    if (ulaz == NULL) {
      fprintf(stderr, "-1\n");
68
      exit(EXIT_FAILURE);
    /* Ucitava se broj linija. */
    fscanf(ulaz, "%d", &n);
    /* Alociranje memorije na osnovu ucitanog broja linija. */
    linije = alociranje_memorije(n);
    /* U slucaju neuspesne alokacije ispisuje se -1 na stderr i
       program zavrsava. */
78
    if (linije == NULL) {
      fprintf(stderr, "-1\n");
80
      exit(EXIT_FAILURE);
82
    /* Iz datoteke se ucita svih n linija. */
84
    for (i = 0; i < n; i++) {
      fscanf(ulaz, "%s", linije[i]);
86
88
    /* Ispisu se u odgovarajucem poretku ucitane linije koje
       zadovoljavaju kriterijum. */
90
    for (i = n - 1; i \ge 0; i--) {
      if (isupper(linije[i][0])) {
        printf("%s\n", linije[i]);
94
    /* Oslobadja se memorija koja je dinamicki alocirana. */
96
    linije = oslobadjanje_memorije(linije, n);
```

```
/* Zatvara se datoteka. */
fclose(ulaz);

exit(EXIT_SUCCESS);
}
```

Napomena: Rešenje koristi biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

#### main.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "stabla.h"
  int sumiraj_n(Cvor * koren, int n)
    /* Ako je stablo prazno, suma je nula */
   if (koren == NULL)
     return 0;
    /* Inace ... */
    /* Ako je n jednako nula, vraca se broj iz korena */
    if (n == 0)
     return koren->broj;
    /* Inace, izracunava se suma na (n-1)-om nivou u levom podstablu,
14
      kao i suma na (n-1)-om nivou u desnom podstablu i vraca se zbir
      te dve izracunate vrednosti jer predstavlja zbir svih cvorova na
16
      n-tom nivou u pocetnom stablu */
    return sumiraj_n(koren->levo, n - 1)
        + sumiraj_n(koren->desno, n - 1);
20 }
22 int main()
    Cvor *koren = NULL;
    int n;
26
   int nivo;
    /* Ucitava se vrednost nivoa */
28
    scanf("%d", &nivo);
    while (1) {
30
      scanf("%d", &n);
      /* Ukoliko je korisnik uneo 0, prekida se dalje citanje. */
      if (n == 0)
        break;
34
      /* Ako nije, dodaje se procitani broj u stablo. */
      if (dodaj_u_stablo(&koren, n) == 1) {
36
        fprintf(stderr, "-1\n");
```

```
oslobodi_stablo(&koren);
    exit(EXIT_FAILURE);

/* Ispisuje se rezultat rada trazene funkcije */
printf("%d\n", sumiraj_n(koren, nivo));

/* Oslobadja se memorija */
oslobodi_stablo(&koren);

exit(EXIT_SUCCESS);
}
```

```
#include <stdio.h>
2 #include <stdlib.h>
  #define MAKS 50
  /* Funkcija ucitava elemenate matrice sa standardnog ulaza */
  void ucitaj_matricu(int m[][MAKS], int v, int k)
    int i, j;
    for (i = 0; i < v; i++) {
      for (j = 0; j < k; j++) {
        scanf("%d", &m[i][j]);
    }
14 }
16 /* Funkcija racuna broj negativnih elemenata u k-oj koloni matrice m
     koja ima v vrsta */
18 int broj_negativnih_u_koloni(int m[][MAKS], int v, int k)
    int broj_negativnih = 0;
    int i;
    for (i = 0; i < v; i++) {
      if (m[i][k] < 0)
        broj_negativnih++;
    return broj_negativnih;
26
  /* Funkcija vraca indeks kolone matrice m u kojoj se nalazi najvise
    negativnih elemenata */
  int maks_indeks(int m[][MAKS], int v, int k)
    int j;
    int broj_negativnih;
34
    /* Inicijalizacija na nulu indeksa kolone sa maksimalnim brojem
```

```
36
       negativnih (maks_indeks_kolone), kao i maksimalnog broja
       negativnih elemenata u nekoj koloni (maks_broj_negativnih) */
    int maks_indeks_kolone = 0;
38
    int maks_broj_negativnih = 0;
40
    for (j = 0; j < k; j++) {
      /* Racuna se broj negativnih u j-oj koloni */
42
      broj_negativnih = broj_negativnih_u_koloni(m, v, j);
      /* Ukoliko broj negativnih u j-toj koloni veci od trenutnog
44
         maksimalnog, azurira se trenutni maksimalni broj negativnih
         kao i indeks kolone sa maksimalnim brojem negativnih */
46
      if (maks_broj_negativnih < broj_negativnih) {</pre>
        maks_indeks_kolone = j;
48
        maks_broj_negativnih = broj_negativnih;
    }
    return maks_indeks_kolone;
54
  int main()
56
    int m[MAKS][MAKS];
    int v, k;
58
    /* Ucitava se broj vrsta matrice */
    scanf("%d", &v);
    /* Proverava se validnost broja vrsta */
    if (v < 0 \mid \mid v > MAKS) {
64
      fprintf(stderr, "-1\n");
      exit(EXIT_FAILURE);
68
    /* Ucitava se broj kolona matrice */
    scanf("%d", &k);
    /* Proverava se validnost broja kolona */
72
    if (k < 0 | | k > MAKS) {
      fprintf(stderr, "-1\n");
74
      exit(EXIT_FAILURE);
    /* Ucitaju se elementi matrice */
    ucitaj_matricu(m, v, k);
    /* Pronalazi se kolona koja sadrzi najveci broj negativnih
80
       elemenata i ispisuje se rezultat */
    printf("%d\n", maks_indeks(m, v, k));
82
    exit(EXIT_SUCCESS);
84
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #define MAKS 128
  int main(int argc, char **argv)
    FILE *f;
    int brojac = 0;
    char linija[MAKS], *p;
    /* Provera da li je broj argumenata komandne linije 3 */
    if (argc != 3) {
      fprintf(stderr, "-1\n");
14
      exit(EXIT_FAILURE);
    /* Otvara se datoteka ciji je naziv zadat kao argument komandne
       linije */
18
    if ((f = fopen(argv[1], "r")) == NULL) {
      fprintf(stderr, "-1\n");
20
      exit(EXIT_FAILURE);
    /* Cita se sadrzaj otvorene datoteke, liniju po liniju. */
    while (fgets(linija, MAKS, f) != NULL) {
24
      p = linija;
      while (1) {
26
        p = strstr(p, argv[2]);
28
        /* Ukoliko nije podniska tj. p je NULL izlazi se iz petlje */
        if (p == NULL)
30
          break;
         /* Inace se uvecava brojac */
         brojac++;
         /* p se pomera da bi se u sledecoj iteraciji posmatra ostatak
34
            linije nakon uocene podniske */
        p = p + strlen(argv[2]);
36
38
    /* Zatvara se datoteka */
40
    fclose(f);
42
    /* Ispisuje se vrednost brojaca */
    printf("%d\n", brojac);
44
    exit(EXIT_SUCCESS);
46
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include <math.h>
  /* Struktura trougao */
6 typedef struct _trougao {
    double xa, ya, xb, yb, xc, yc;
8 } trougao;
10 /* Funkcija racuna duzinu duzi */
  double duzina(double x1, double y1, double x2, double y2)
    return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
14 }
16 /* Funkcija racuna povrsinu trougla koristeci Heronov obrazac */
  double povrsina(trougao t)
    /* Racunaju se duzine stranica trougla */
20
   double a = duzina(t.xb, t.yb, t.xc, t.yc);
    double b = duzina(t.xa, t.ya, t.xc, t.yc);
   double c = duzina(t.xa, t.ya, t.xb, t.yb);
    /* Racuna se poluobim trougla */
    double s = (a + b + c) / 2;
    /* Primenom Heronovog obrasca racuna se povrsina trougla */
    return sqrt(s * (s - a) * (s - b) * (s - c));
  }
28
  /* Funkcija poredi dva trougla: ukoliko je povrsina trougla koji je
     prvi argument funkcije manja od povrsine trougla koji je drugi
30
     element funkcije funcija vraca 1, ukoliko je veca -1, a ukoliko
     su povrsine dva trougla jednake vraca nulu. Dakle, funkcija je
     napisana tako da se moze proslediti funkciji qsort da se niz
     trouglova sortira po povrsini opadajuce. */
  int poredi(const void *a, const void *b)
36 | €
    trougao x = *(trougao *) a;
    trougao y = *(trougao *) b;
38
    double xp = povrsina(x);
    double yp = povrsina(y);
40
    if (xp < yp)
     return 1;
42
    if (xp > yp)
     return -1;
44
    return 0;
46 }
48 int main()
50
  FILE *f;
```

```
int n, i;
    trougao *niz;
    /* Otvara se datoteka ciji je naziv trouglovi.txt */
54
    if ((f = fopen("trouglovi.txt", "r")) == NULL) {
      fprintf(stderr, "-1\n");
56
      exit(EXIT_FAILURE);
58
    /* Ucitava se podatak o broju trouglova iz datoteke */
60
    if (fscanf(f, "%d", &n) != 1) {
      fprintf(stderr, "-1\n");
62
      exit(EXIT_FAILURE);
64
    /* Dinamicka alokacija memorije: za niz trouglova duzine n
66
       rezervise se memorijski prostor */
    if ((niz = malloc(n * sizeof(trougao))) == NULL) {
68
      fprintf(stderr, "-1\n");
      exit(EXIT_FAILURE);
    /* Ucitavaju se podaci u niz iz otvorene datoteke */
    for (i = 0; i < n; i++) {
74
      if (fscanf(f, "%lf%lf%lf%lf%lf", &niz[i].xa, &niz[i].ya,
                  &niz[i].xb, &niz[i].yb, &niz[i].xc,
76
                  &niz[i].yc) != 6) {
        fprintf(stderr, "-1\n");
        exit(EXIT_FAILURE);
80
82
    /* Poziva se funkcija qsort da sortira niz na osnovu funkcije
       poredi */
84
    qsort(niz, n, sizeof(trougao), &poredi);
86
    /* Ispisuje se sortirani niz na standardni izlaz */
    for (i = 0; i < n; i++)
88
      printf("%g %g %g %g %g %g\n", niz[i].xa, niz[i].ya, niz[i].xb,
             niz[i].yb, niz[i].xc, niz[i].yc);
90
    /* Oslobadja se dinamicki alocirana memorija */
92
    free(niz);
94
    /* Zatvara se datoteka */
    fclose(f);
96
    exit(EXIT_SUCCESS);
```

Napomena: Rešenje koristi biblioteku za rad sa binarnim pretraživačkim stablima iz zadatka 4.14.

main.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "stabla.h"
  /* Funkcija ucitava brojeve sa standardnog ulaza i smesta ih u
     stablo. Funkcija vraca 1 u slucaju neuspesnog dodavanja elementa
     u stablo, a inace 0. */
8 int ucitaj_stablo(Cvor ** koren)
    *koren = NULL;
    int x;
    /* Sve dok ima brojeva na standardnom ulazu, ucitani brojevi se
12
       dodaju u stablo. Ukoliko funkcija dodaj_u_stablo vrati 1, onda
       je i povratna vrednost iz funkcije ucitaj_stablo 1. */
14
    while (scanf("%d", &x) == 1)
      if (dodaj_u_stablo(koren, x) == 1)
        return 1;
    return 0;
18
  }
20
  /* Funkcija prebrojava broj cvorova na n-tom nivou u stablu */
22 int prebroj_n(Cvor * koren, int n)
    /* Ukoliko je stablo prazno, rezultat je nula. Takodje, ako je n
       negativan broj, na tom nivou nema cvorova (rezultat je nula). */
    if (koren == NULL || n < 0)
      return 0;
    /* Ukoliko je n = 0, na tom nivou je samo koren. Ukoliko ima
28
       jednog potomka funkcija vraca 1, inace 0 */
30
    if (n == 0) {
      if (koren->levo == NULL && koren->desno != NULL)
      if (koren->levo != NULL && koren->desno == NULL)
        return 1;
34
      return 0;
    }
36
    /* Broj cvorova na n-tom nivou je jednak zbiru broja cvorova na
       (n-1)-om nivou levog podstabla i broja cvorova na (n-1)-om
38
       nivou levog podstabla */
    return prebroj_n(koren->levo, n - 1)
40
        + prebroj_n(koren->desno, n - 1);
42 }
44 int main()
```

```
Cvor *koren;
    int n;
    scanf("%d", &n);
48
    /* Ucitavaju se elementi u stablo. U slucaju neuspesne alokacije
       oslobodja se alocirana memorija i izlazi se iz programa. */
    if (ucitaj_stablo(&koren) == 1) {
      fprintf(stderr, "-1\n");
      oslobodi_stablo(&koren);
54
      exit(EXIT_FAILURE);
56
    /* Ispisuje se rezultat */
58
    printf("%d\n", prebroj_n(koren, n));
60
    /* Oslobadja se dinamicki alocirana memorija za stablo */
    oslobodi_stablo(&koren);
62
    exit(EXIT_SUCCESS);
64
```

```
#include <stdio.h>
  /* Funkcija vraca broj ciji binarni zapis se dobija kada se binarni
     zapis argumenta x rotira za n mesta udesno */
  unsigned int rotiraj(unsigned int x, unsigned int n)
    int i;
    unsigned int maska = 1;
    /* Formira se maska sa n jedinica na kraju, npr za n=4 maska bi
       izgledala: 000...00001111 */
    /* Maska se moze formirati i naredbom: maska = (1 << n) - 1; U
       nastavku je drugi nacin. */
    for (i = 1; i < n; i++)
      maska = (maska << 1) | 1;
    /* Kada se x poremeri za n mesta udesno x >> n, poslednjih n
       bitova binarne reprezentacije broja x ce "ispasti". Za rotaciju
       je potrebno da se tih n bitova postavi na pocetak broja.
18
       Kreirana maska omogucava da se tih n bitova izdvoji sa (maska &
       x), a zatim se pomeranjem za (sizeof(unsigned) * 8 - n) mesta
       ulevo tih n bitova postavlja na pocetak. Primenom logicke
       disjunkcije dobija se rotirani broj. */
    return (x >> n) \mid ((maska & x) << (size of (unsigned) * 8 - n));
24
  int main()
26 {
    unsigned int x, n;
```

```
/* Ucitaju se brojevi sa standardnog ulaza */
scanf("%u%u", &x, &n);

/* Ispisuje se rezultat */
printf("%u\n", rotiraj(x, n));
return 0;
}
```

#### liste.h

```
#ifndef _LISTE_H_
2 #define _LISTE_H_ 1
4 /* Struktura koja predstavlja cvor liste */
  typedef struct cvor {
   double vrednost;
    struct cvor *sledeci;
  } Cvor;
/* Pomocna funkcija koja kreira cvor. */
  Cvor *napravi_cvor(double broj);
12
  /* Funkcija oslobadja dinamicku memoriju zauzetu za elemente liste
     ciji se pocetni cvor nalazi na adresi adresa_glave. */
14
  void oslobodi_listu(Cvor ** adresa_glave);
  /* Funkcija pronalazi i vraca pokazivac na poslednji element liste,
     ili NULL kao je lista prazna */
18
  Cvor *pronadji_poslednji(Cvor * glava);
20
  /* Funkcija dodaje novi cvor na kraj liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije,inace vraca 0. */
22
  int dodaj_na_kraj_liste(Cvor ** adresa_glave, double broj);
24
  /* Funkcija prikazuje sve elemente liste pocev od glave ka kraju
     liste. */
26
  void ispisi_listu(Cvor * glava);
28
  #endif
```

#### liste.c

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

```
5 /* Pomocna funkcija koja kreira cvor. */
  Cvor *napravi_cvor(double broj)
  {
    Cvor *novi = (Cvor *) malloc(sizeof(Cvor));
    if (novi == NULL)
     return NULL:
    /* inicijalizacija polja u novom cvoru */
    novi->vrednost = broj;
    novi->sledeci = NULL;
13
    return novi;
15 }
  /* Funkcija oslobadja dinamicku memoriju zauzetu za elemente liste
     ciji se pocetni cvor nalazi na adresi adresa_glave. */
  void oslobodi_listu(Cvor ** adresa_glave)
19
    Cvor *pomocni = NULL;
    while (*adresa_glave != NULL) {
      pomocni = (*adresa_glave)->sledeci;
      free(*adresa_glave);
      *adresa_glave = pomocni;
  }
  /* Funkcija pronalazi i vraca pokazivac na poslednji element liste,
    ili NULL kao je lista prazna */
  Cvor *pronadji_poslednji(Cvor * glava)
31
    /* Ako je lista prazna, nema ni poslednjeg cvor i u tom slucaju
       funkcija vraca NULL. */
    if (glava == NULL)
     return NULL;
    while (glava->sledeci != NULL)
      glava = glava->sledeci;
39
    return glava;
41
  /* Funkcija dodaje novi cvor na kraj liste. Vraca 1 ukoliko je bilo
     greske pri alokaciji memorije, inace vraca 0. */
43
  int dodaj_na_kraj_liste(Cvor ** adresa_glave, double broj)
45
    Cvor *novi = napravi_cvor(broj);
    if (novi == NULL)
47
      return 1;
    if (*adresa_glave == NULL) {
49
      *adresa_glave = novi;
      return 0;
    Cvor *poslednji = pronadji_poslednji(*adresa_glave);
    poslednji->sledeci = novi;
    return 0;
```

```
/* Funkcija prikazuje elemente liste pocev od glave ka kraju liste.

*/
void ispisi_listu(Cvor * glava)
{
  for (; glava != NULL; glava = glava->sledeci)
    printf("%.21f ", glava->vrednost);
  putchar('\n');
}
```

main.c

```
#include <stdio.h>
2 #include <stdlib.h>
  #include "liste.h"
  /* Funkcija umece novi cvor iza tekuceg u listi */
6 void dodaj_iza(Cvor * tekuci, Cvor * novi)
    /* Novi cvor se dodaje iza tekuceg cvora. */
    novi->sledeci = tekuci->sledeci;
    tekuci->sledeci = novi;
  }
12
  /* Funkcija koja dopunjuje listu na nacin opisan u tekstu zadatka.
     Vraca 1 ukoliko je bilo greske pri alokaciji memorije, inace
14
     vraca 0. */
int dopuni_listu(Cvor ** adresa_glave)
    Cvor *tekuci;
18
    Cvor *novi;
    double aritmeticka_sredina;
    /* U slucaju prazne ili jednoclane liste, funkcija vraca 1 */
    if (*adresa_glave == NULL || (*adresa_glave)->sledeci == NULL)
      return 1:
24
    /* Promenljiva tekuci se inicijalizacuje da pokazuje na pocetni
       cvor */
26
    tekuci = *adresa_glave;
    /* Sve dok ima cvorova u listi racuna se aritmeticka sredina
       vrednosti u susednim cvorovima i kreira cvor sa tom vrednoscu.
       U slucaju neuspele alokacije novog cvora, funkcija vraca 1.
       Inace, novi cvor se umece izmedju dva cvora za koje je racunata
30
       aritmeticka sredina */
    while (tekuci->sledeci != NULL) {
      aritmeticka_sredina =
          ((tekuci)->vrednost + ((tekuci)->sledeci)->vrednost) / 2;
34
      novi = napravi_cvor(aritmeticka_sredina);
      if (novi == NULL)
36
      /* Poziva se funkcija koja umece novi cvor iza tekuceg cvora */
38
      dodaj_iza(tekuci, novi);
```

```
/* Tekuci cvor se pomera na narednog u listi (to je novoumetnuti
40
         cvor), a zatim jos jednom da bi pokazivao na naredni cvor iz
         polazne liste */
42
      tekuci = tekuci->sledeci;
      tekuci = tekuci->sledeci;
44
    return 0;
46
48
  int main()
  {
    Cvor *glava = NULL;
    double broj;
    /* Dok se ne stigne do kraja ulaza, ucitavaju se elementi i dodaju
54
       se na kraj liste */
    while (scanf("%lf", &broj) > 0) {
56
      /* Ako je funkcija vratila 1, onda je bilo greske pri alokaciji
         memorije za nov cvor. Memoriju alociranu za cvorove liste
58
         treba osloboditi. */
      if (dodaj_na_kraj_liste(&glava, broj) == 1) {
        fprintf(stderr, "Greska: Neuspela alokacija za cvor %lf.\n",
                broj);
        oslobodi_listu(&glava);
        exit(EXIT_FAILURE);
64
    }
    /* Poziva se funkcija da dopuni listu. Ako je funkcija vratila 1,
       onda je bilo greske pri alokaciji memorije za nov cvor.
       Memoriju alociranu za cvorove liste treba osloboditi. */
    if (dopuni_listu(&glava) == 1) {
      fprintf(stderr, "Greska: Neuspela alokacija za cvor %lf.\n",
              broj);
74
      oslobodi_listu(&glava);
      exit(EXIT_FAILURE);
76
    /* Ispisuju se elementi liste */
    ispisi_listu(glava);
80
    /* Oslobadja se memorija rezervisana za listu */
    oslobodi_listu(&glava);
    exit(EXIT_SUCCESS);
84
```

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

```
| #include "matrica.h"
  /* Funkcija racuna zbir elemenata v-te vrste */
6 int zbir_vrste(int **M, int n, int v)
    int j, zbir = 0;
8
   for (j = 0; j < n; j++)
     zbir += M[v][j];
    return zbir;
12 }
14 /* Funkcija racuna zbir elemenata k-te kolone */
  int zbir_kolone(int **M, int n, int k)
16 {
    int i, zbir = 0;
  for (i = 0; i < n; i++)
18
     zbir += M[i][k];
   return zbir;
20
  }
  /* Funkcija proverava da li je kvadrat koji joj se prosledjuje kao
     argument magican. Ukoliko jeste magican funkcija vraca 1, inace
     0. Argument funkcije zbir ce sadrzati zbir elemenata neke vrste
     ili kolone ukoliko je kvadrat magican. */
26
  int magicni_kvadrat(int **M, int n, int *zbir_magicnog)
28 {
    int i, j;
   int zbir = 0, zbir_pom;
30
    /* Promenljivu zbir inicijalizujemo na zbir 0-te vreste */
    zbir = zbir_vrste(M, n, 0);
    /* Racunaju se zbirovi u ostalim vrstama i ako neki razlikuje od
34
       vrednosti promeljive zbir funkcija vraca 1 */
    for (i = 1; i < n; i++) {
36
      zbir_pom = zbir_vrste(M, n, i);
      if (zbir_pom != zbir)
38
        return 0;
    }
40
    /* Racunaju se zbirovi u svim kolonama i ako neki razlikuje od
       vrednosti promeljive zbir funkcija vraca 1 */
42
    for (j = 0; j < n; j++) {
     zbir_pom = zbir_kolone(M, n, j);
44
      if (zbir_pom != zbir)
        return 0;
46
    }
    /* Inace su zbirovi svih vrsta i kolona jednaki, postavlja se
48
       vresnost u zbir_magicnog i funkcija vraca 1 */
    *zbir_magicnog = zbir;
    return 1;
52 }
54 int main()
```

```
int n;
    int **matrica = NULL;
    int zbir_magicnog;
    scanf("%d", &n);
60
    /* Provera da li je n strogo pozitivan */
    if (n \le 0) {
62
      printf("-1\n");
      exit(EXIT_FAILURE);
64
66
    /* Dinamicka alokacija kvadratne matrice dimenzije n */
    matrica = alociraj_matricu(n, n);
    if (matrica == NULL) {
      printf("-1\n");
      exit(EXIT_FAILURE);
    /* Ucitavaju se elementi matrice sa standardnog ulaza */
74
    ucitaj_matricu(matrica, n, n);
    /* Ispisuje se rezultat na osnovu fukcije magicni_kvadrat */
    if (magicni_kvadrat(matrica, n, &zbir_magicnog)) {
78
      printf("%d\n", zbir_magicnog);
    } else
80
      printf("-\n");
82
    /* Oslobadja se dinamicki alocirana memorija */
    matrica = dealociraj_matricu(matrica, n);
84
    exit(EXIT_SUCCESS);
86
```

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

#define BITOVA_U_BAJTU 8

/* Funkcija u datom broju x menja mesta prvom i četvrtom bajtu */
unsigned int zamena(unsigned int x)

{
    /* Deklaracija promenljivih za odgovarajuce maske i pomocne
    promenljive */
    unsigned maska_prvi_bajt, maska_cetvrti_bajt;
unsigned maska_prvi_bajt_komplement, maska_cetvrti_bajt_komplement;
unsigned prvi_bajt, cetvrti_bajt;
unsigned i;
```

```
/* Maska prvi bajt odgovara broju cija je binarna reprezentacija
16
       00000....00000111111111 (8 bitova najmanje tezine su jedinice, a
       ostalo su nule) moze se dobiti i tako sto se maska_prvi_bajt
18
       postavi na heksadekadnu vrednost FF. Drugi nacin za
       inicijalizaciju maske maska_prvi_bajt je dodavanjem jedinica sa
20
       desne strane: */
    maska_prvi_bajt = 1;
    for (i = 1; i < BITOVA_U_BAJTU; i++)
      maska_prvi_bajt = maska_prvi_bajt << 1 | 1;</pre>
24
    /* Maska cetvrti bajt odgovara broju cija je binarna
26
       reprezentacija 1111111100000....00000 (8 bitova najvece tezine
       su jedinice, a ostalo su nule) i moze se dobiti pomeranjem
28
       bitova prethodno kreirane maske maska_prvi_bajt tako da
       jedinice budu na poziciji bajta najvece tezine. */
30
    maska_cetvrti_bajt =
        maska_prvi_bajt << ((sizeof(unsigned) - 1) * BITOVA_U_BAJTU);</pre>
    /* Primenom operatora ~ na maska_prvi_bajt dobija se broj cija je
34
       binarna reprezentacija 11111....1111100000000 (8 bitova
       najmanje tezine su nule, a ostalo su jedinice) */
36
    maska_prvi_bajt_komplement = ~maska_prvi_bajt;
    /* Primenom operatora ~ na maska_prvi_bajt dobija se broj cija je
38
       binarna reprezentacija 000000011111....11111 (8 bitova najvece
       tezine su nule, a ostalo su jedinice) */
40
    maska_cetvrti_bajt_komplement = ~maska_cetvrti_bajt;
42
    /* U promenljivu prvi_bajt se smesta broj koji se dobija kada se
       bitovi prvog bajta broja x pomere ulevo, tako da budu na
44
       poziciji cetvrtog bajta */
    prvi_bajt =
46
        (maska_prvi_bajt & x) << ((sizeof(unsigned) - 1) *</pre>
                                   BITOVA_U_BAJTU);
48
    /* U promenljivu cetvrti_bajt se smesta broj koji se dobija kada
       se bitovi cetvrtog bajta broja x pomere udesno, tako da budu na
       poziciji prvog bajta */
52
    cetvrti_bajt =
        (maska_cetvrti_bajt & x) >> ((sizeof(unsigned) - 1) *
                                      BITOVA_U_BAJTU);
54
    /* Na nule se postavlja 8 bitova najmanje tezine, a ostali bitovi
       ostaju nepromenjeni */
    x = x & maska_prvi_bajt_komplement;
    /* Na nule se postavlja 8 bitova najvece tezine, a ostali bitovi
       ostaju nepromenjeni */
    x = x & maska_cetvrti_bajt_komplement;
62
    /* Na bitove na poziciji cetvrtog bajta se postavljaju bitovi iz
64
       prvog bajta */
    x = x | prvi_bajt;
```

```
/* Na bitove na poziciji cetvrtog bajta se postavljaju bitovi iz
       prvog bajta */
    x = x | cetvrti_bajt;
    return x;
72
  }
74
  int main()
  {
76
    int x;
78
    /* Sa standardnog ulaza se ucitava ceo broj */
    scanf("%d", &x);
80
    /* Provera da li je uneti broj negativan */
82
    if (x < 0) {
      fprintf(stderr, "-1\n");
84
      exit(EXIT_FAILURE);
86
    /* Ispisuje se rezultat primene funkcije zamena na uneti broj x */
88
    printf("%u\n", zamena(x));
90
    exit(EXIT_SUCCESS);
  }
92
```

```
#include <stdio.h>
  #include <stdlib.h>
  #include "stabla.h"
  /* Funkcija racuna duzinu najduzeg puta od korena do nekog lista */
  int najduzi_put(Cvor * koren)
    /* Pomocne promenljive */
    int najduzi_u_levom, najduzi_u_desnom;
    /* Ako je stablo prazno, povratna vrednost je -1 */
    if (koren == NULL)
12
     return -1;
14
    /* Rekurzivno se izracuna duzina najduzeg puta u levom podstablu */
    najduzi_u_levom = najduzi_put(koren->levo);
16
    /* Rekurzivno se izracuna duzina najduzeg puta u levom podstablu */
18
    najduzi_u_desnom = najduzi_put(koren->desno);
20
```

```
/* Veca od prethodno izracunatih vrednosti za podstabla se uvecava
       za 1 i vraca kao konacan rezultat */
    return 1 + (najduzi_u_levom >
                najduzi_u_desnom ? najduzi_u_levom : najduzi_u_desnom);
24
26
  int main()
28 | {
    Cvor *stablo = NULL;
30
    int x;
    /* U svakoj iteraciji se procitani broj dodaje u stablo. */
    while (scanf("%d", &x) == 1)
      if (dodaj_u_stablo(&stablo, x) == 1) {
        fprintf(stderr, "-1\n");
        exit(EXIT_FAILURE);
36
38
    /* Ispisuje se rezultat rada trazene funkcije */
    printf("%d\n", najduzi_put(stablo));
40
    /* Oslobadja se memorija */
42
    oslobodi_stablo(&stablo);
44
    exit(EXIT_SUCCESS);
  }
46
```

```
#include <stdio.h>
  #include <stdlib.h>
3 /* Ime datoteke nije duze od 30 karaktera */
  #define MAX 31
  /* Funkcija alocira memorijski prostor za matricu sa n vrsta i m
    kolona */
  float **alociraj_matricu(int n, int m)
   float **matrica = NULL;
   int i, j;
    /* Alocira se prostor za niz vrsta matrice */
    matrica = (float **) malloc(n * sizeof(float *));
    /* Ako alokacija nije prosla uspesno, povratna vrednost funkcije
       ce biti NULL, sto mora biti provereno u main funkciji */
    if (matrica == NULL)
17
      return NULL;
19
    /* Alocira se prostor za svaku vrstu matrice */
    for (i = 0; i < n; i++) {
      matrica[i] = (float *) malloc(m * sizeof(float));
```

```
/* Ako alokacija nije prosla uspesno, oslobadjaju se svi
23
         prethodno alocirani resursi, i povratna vrednost je NULL */
      if (matrica[i] == NULL) {
        for (j = 0; j < i; j++)
          free(matrica[j]);
27
        free(matrica);
        return NULL;
29
31
    return matrica;
35
  /* Funkcija oslobadja alocirani memorijski prostor */
  float **dealociraj_matricu(float **matrica, int n)
37
    int i;
39
    /* Oslobadja se prostor rezervisan za svaku vrstu */
    for (i = 0; i < n; i++)
41
      free(matrica[i]);
43
    /* Oslobadja se memorija za niz pokazivaca na vrste */
    free(matrica);
45
    /* Matrica postaje prazna, tj. nealocirana */
    return NULL;
  }
49
  /* Funkcija prebrojava koliko se puta pojavljuje broj x u i-toj
     vrsti matrice A, gde je m broj elemenata u vrsti */
  int prebroj_u_itoj_vrsti(float **A, int i, int m, int x)
    int j;
    int broj = 0;
    for (j = 0; j < m; j++) {
      if (A[i][j] == x)
59
        broj++;
    return broj;
61
63
  /* Funkcija vraca indeks vrste matrice A u kojoj se realan broj x
     pojavljuje najmanje puta */
65
  int indeks_vrste(float x, float **A, int n, int m)
67
    /* Indeks vrste sa minimalnim brojem pojavljivanja broja x */
    int min;
69
    /* Broj pojavljivanja broja x u vrsti sa indeksom min */
    int min_broj;
    /* Promenljiva u kojoj ce se racunati broj pojavljivanja broja x u
       tekucnoj vrsti */
    int broj_u_vrsti;
```

```
75
     /* Pomocne promenljive */
     int i;
     /* Promenljiva min se inicijalizuje na nulu, a min_broj na broj
        pojavljivanja broja x u nultoj vrsti */
79
     min = 0:
     min_broj = prebroj_u_itoj_vrsti(A, 0, m, x);
81
     /* Za svaku vrstu (osim nulte) se racuna broj pojavljivanja broja
83
        x u njoj, pa ukoliko je taj broj manji od trenutno najmanjeg
        azuriraju se promenljive min i min_broj */
85
     for (i = 1; i < n; i++) {
       broj_u_vrsti = prebroj_u_itoj_vrsti(A, i, m, x);
87
       if (broj_u_vrsti < min_broj) {</pre>
         min_broj = broj_u_vrsti;
89
         min = i;
       }
91
     }
     /* Funkcija vraca odgovarajuci indeks vrste */
     return min;
95
97
   int main()
99 {
     FILE *in;
    char datoteka[MAX];
    float broj;
    float **A = NULL;
    int i, j, m, n;
     /* Sa standardnog ulaza se ucitava ime datoteke i realan broj */
     scanf("%s", datoteka);
     scanf("%f", &broj);
     /* Otvara se datoteka za citanje */
     in = fopen(datoteka, "r");
     /* Provera da li je datoteka uspesno otvorena */
113
     if (in == NULL) {
       fprintf(stderr, "-1\n");
       exit(EXIT_FAILURE);
     }
117
     /* Dimenzije matrice se ucitavaju iz datoteke (prva dva cela broja
119
        u datoteci). U slucaju neuspesnog ucitavanja, na standardni
        izlaz za greske se ispisuje -1 i prekida se program. */
     if (fscanf(in, "%d %d", &n, &m) == EOF) {
       fprintf(stderr, "-1\n");
123
       exit(EXIT_FAILURE);
     }
```

```
/* Provera da li su ucitani brojevi m i n pozitivni */
     if (n <= 0 || m <= 0) {
       fprintf(stderr, "-1\n");
       exit(EXIT_FAILURE);
     /* Alokacija matrice */
     A = alociraj_matricu(n, m);
     /* Provera da li je alokacija uspela */
     if (A == NULL) {
       fprintf(stderr, "-1\n");
       exit(EXIT_FAILURE);
141
     /* Ucitavaju se elementi matrice iz datoteke */
     for (i = 0; i < n; i++) {
143
       for (j = 0; j < m; j++)
         fscanf(in, "%f", &A[i][j]);
145
147
     /* Zatvara se datoteka */
     fclose(in);
149
     /* Ispisuje se rezultat poziva funkcije */
     printf("%d\n", indeks_vrste(broj, A, n, m));
153
     /* Oslobadja se memorija koju je zauzimala matrica */
     A = dealociraj_matricu(A, n);
     exit(EXIT_SUCCESS);
```