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, crteži i korice: *autori*.
Štampa: Copy Centar, Beograd. Tiraž 200.

CIP 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]

a) Програмирање - Задаци b) Програмски језик "C"- Задаци
COBISS.SR-ID 221508876

Sadržaj

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[1].

Autori

[1] Adrese autora su: `milena`, `jgraovac`, `nina`, `aspasic`, `mirko`, `andjelkaz`, sa nastavkom `@matf.bg.ac.rs`

1

Pokazivači

1.1 Pokazivačka aritmetika

Zadatak 1.1 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 ($0 < n \leq 100$), 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:
 Unesite dimenziju niza:  3
 Unesite elemente niza:
 1 -2 3
 Nakon obrtanja elemenata, niz je:
 3 -2 1
 Nakon ponovnog obrtanja elemenata,
 niz je:
 3 -2 1
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju niza:  0
Izlaz za greške:
 Greska: Neodgovarajuca dimenzija niza.
```

Zadatak 1.2 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 ($0 < n \leq 100$) realnog niza, a zatim i elemente niza. Na standardni izlaz ispisati zbir, proizvod, minimalni i maksimalni element učitanog niza.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  3
 Unesite elemente niza:
-1.1 2.2 3.3
 Zbir elemenata niza je 4.400.
 Proizvod elemenata niza je -7.986
 Minimalni element niza je -1.100
 Maksimalni element niza je 3.300
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  5
 Unesite elemente niza:
 1.2 3.4 0.0 -5.4 2.1
 Zbir elemenata niza je 1.300.
 Proizvod elemenata niza je -0.000.
 Minimalni element niza je -5.400.
 Maksimalni element niza je 3.400.
```

Zadatak 1.3 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 ($0 < n \leq 100$) celobrojnog niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene napisane funkcije nad učitanim nizom.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  5
 Unesite elemente niza:
 1 2 3 4 5
 Transformisan niz je:
 2 3 3 3 4
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  4
 Unesite elemente niza:
 4 -3 2 -1
 Transformisan niz je:
 5 -2 1 -2
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  0
IZLAZ ZA GREŠKE:
 Greska: Neodgovarajuca dimenzija niza.
```

Primer 4

```
INTERAKCIJA SA PROGRAMOM:
 Unesite dimenziju niza:  101
IZLAZ ZA GREŠKE:
 Greska: Neodgovarajuca dimenzija niza.
```

Zadatak 1.4 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 tražiti da izabere koje od ova dva rešenja treba koristiti prilikom ispisa.

Primer 1

```
POKRETANJE: ./a.out prvi 2. treci -4

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

Primer 2

```
POKRETANJE: ./a.out

INTERAKCIJA SA PROGRAMOM:
 Broj argumenata komandne linije je 1.
 Kako zelite da ispisete argumente?
 Koriscenjem indeksne ili pokazivacke
 sintakse (I ili P)?  P
 Argumenti komandne linije su:
 0 ./a.out
 Pocetna slova argumenata komandne
 linije:
 .
```

Zadatak 1.5 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.
```

Primer 2

```
POKRETANJE: ./a.out

INTERAKCIJA SA PROGRAMOM:
 Broj argumenata
 koji su palindromi je 0.
```

Zadatak 1.6 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.

Primer 1

```
POKRETANJE: ./a.out ulaz.txt 1

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

INTERAKCIJA SA PROGRAMOM:
 Broj reci ciji je broj karaktera 1 je 3.
```

Primer 2

```
POKRETANJE: ./a.out ulaz.txt 2

DATOTEKA ULAZ.TXT NE POSTOJI

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

Primer 3

```
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 1.7 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 opcije program proverava 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 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 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 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
```

1.2 Višedimenzioni nizovi

Zadatak 1.8 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 ($0 < n \leq 100$), a zatim i elemente matrice. Na standardni izlaz ispisati učitanu matricu, a zatim trag, euklidsku normu i vandijagonalnu normu učitane matrice.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:  3
 Unesite elemente matrice po vrstama:
 1 -2 3
 4 -5 6
 7 -8 9
 Trag matrice je 5.
 Euklidska norma matrice je 16.88.
 Vandijagonalna norma matrice je 11.
```

Primer 2

```
Interakcija sa programom:
 Unesite dimenziju matrice:  0
Izlaz za greške:
 Greska: Neodgovarajuca dimenzija
 matrice.
```

Zadatak 1.9 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 ($0 < n \leq 100$), a zatim i elemente matrica. Na standardni izlaz ispisati da li su matrice jednake, a zatim ispisati zbir i proizvod učitanih matrica.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Unesite broj vrsta matrica:  3
Unesite elemente prve matrice po vrstama:
1 2 3
1 2 3
1 2 3
Unesite elemente druge matrice po vrstama:
1 2 3
1 2 3
1 2 3
Matrice su jednake.
Zbir matrica je:
2 4 6
2 4 6
2 4 6
Proizvod matrica je:
6 12 8
6 12 8
6 12 8
```

Zadatak 1.10 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 ($0 < n \leq 64$), a potom i sami elementi matrice. Na standardni izlaz ispisati rezultat testiranja napisanih funkcija.

Primer 1

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

ULAZ.TXT
 4
 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 1.11 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 ($0 < n \leq 32$) zadaje kao argument komandne linije. Na standardni izlaz ispisati rezultat primene prethodno napisanih funkcija.

Primer 1

```
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 1.12 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 ($0 < n \leq 32$), a zatim i njene elemente. Na standardni izlaz ispisati rezultat primene napisane funkcije na učitanu matricu.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta matrice:  4
 Unesite elemente matrice po vrstama:
 1 0 0 0
 0 1 0 0
 0 0 1 0
 0 0 0 1
 Matrica je ortonormirana.
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta matrice:  3
 Unesite elemente matrice po vrstama:
 1 2 3
 5 6 7
 1 4 2
 Matrica nije ortonormirana.
```

Zadatak 1.13 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 ($0 < n \leq 10$) i broj kolona m ($0 < n \leq 10$) matrice, a zatim i njene elemente. Na standardni izlaz spiralno ispisati elemente učitane matrice.

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona
 matrice:
 3 3
 Unesite elemente matrice po vrstama:
 1 2 3
 4 5 6
 7 8 9
 Spiralno ispisana matrica:
 1 2 3 6 9 8 7 4 5
```

Primer 2

```
Interakcija sa programom:
 Unesite broj vrsta i broj kolona
 matrice:
 3 4
 Unesite elemente matrice po vrstama:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 Spiralno ispisana matrica:
 1 2 3 4 8 12 11 10 9 5 6 7
```

Zadatak 1.14 Napisati funkciju koja izračunava k-ti stepen kvadratne matrice dimenzije $n \times n$ ($0 < n \leq 32$). Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati broj vrsta celobrojne matrice n, elemente matrice i stepen k ($0 < k \leq 10$). Na standardni izlaz ispisati rezultat primene napisane funkcije. Napomena: *Voditi računa da se prilikom stepenovanja matrice izvrši što manji broj množenja.*

Primer 1

```
Interakcija sa programom:
 Unesite broj vrsta kvadratne matrice:  3
 Unesite elemente matrice po vrstama:
 1 2 3
 4 5 6
 7 8 9
 Unesite stepen koji se racuna:  8
 8. stepen matrice je:
 510008400 626654232 743300064
 1154967822 1419124617 1683281412
 1799927244 2211595002 2623262760
```

1.3 Dinamička alokacija memorije

Zadatak 1.15 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.

Primer 1

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

(a) realokaciju memorije niza vršiti korišćenjem `malloc()` funkcije,

(b) realokaciju memorije niza vršiti korišćenjem `realloc()` funkcije.

Od korisnika 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 -1 5 -2 4 -3 0
 Niz u obrnutom poretku je:
 3 4 -2 5 -1 6
```

Zadatak 1.17 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 neće biti duže od 50 karaktera i da neće sadržati 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 1.18 Napisati program koji sa standardnog ulaza učitava matricu realnih brojeva. Prvo se učitavaju broj vrsta n i broj kolona m matrice, a zatim i elementi matrice. Na standardni izlaz ispisati trag matrice. Ne praviti nikakve pretpostavke o maksimalnoj dimenziji matrice.

Primer 1

```
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 1.19 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 standardni izlaz.

(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 $n \times n$ 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 $n \times m$ 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 $n \times n$ u već otvorenu datoteku `f`. U slučaju neuspešnog upisivanja vratiti vrednost različitu od 0.

Napisati programe koji testiraju napisanu biblioteku.

(1) Sa standardnog ulaza učitati broj vrsta i broj kolona matrice, a zatim i elemente matrice. Nakon toga sadržaj matrice upisati u datoteku *matrica.txt*.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 3
 Unesite broj kolona matrice: 4
 Unesite elemente matrice po vrstama:
 1 2 3 4
 5 6 7 8
 9 10 11 12

MATRICA.TXT
 1 2 3 4
 5 6 7 8
 9 10 11 12
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj vrsta matrice: 5
 Unesite broj kolona matrice: 0
IZLAZ ZA GREŠKE:
 Greska: Broj vrsta i broj kolona ne mogu
 biti negativni brojevi.
```

(2) Program prima kao prvi argument komandne linije putanju do datoteke u kojoj se, redom, nalaze dimenzija i elementi kvadratne matrice. Zatim učitava matricu i ispisuje je na standardni izlaz.

Test 1

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

ULAZ.TXT
 4
 1 2 3 4
 5 6 7 8
 9 10 11 12
 13 14 15 16

IZLAZ:
 1 2 3 4
 5 6 7 8
 9 10 11 12
 13 14 15 16
```

Test 2

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

ULAZ.TXT
 4
 1 2 3 4
 5 6 7 8
 9 10 11 12
 13 14 15 16

IZLAZ ZA GREŠKE:
 Greska: Neispravan pocetak
     datoteke.
```

Test 3

```
POKRETANJE: ./a.out

IZLAZ:
 Koriscenje programa:
 ./a.out datoteka
```

Zadatak 1.20 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 1.19.*

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 1.21 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 1.19.*

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
 najveci zbir.
```

Zadatak 1.22 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

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

MATRICA.TXT
 3
 1.1 -2.2 3.3
 -4.4 5.5 -6.6
 7.7 -8.8 9.9
```

```
INTERAKCIJA SA PROGRAMOM:
 Zbir apsolutnih vrednosti ispod
 sporedne dijagonale je 25.30.
 Transformisana matrica je:
1.10 -1.10 1.65
-8.80 5.50 -3.30
15.40 -17.60 9.90
```

Zadatak 1.23 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

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

MATRICE.TXT
 3
 1.1 -2.2 3.3
 -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
```

```
INTERAKCIJA SA PROGRAMOM:
 1.1 -2.2 3.3
 -1.1 2.2 -3.3
 -4.4 5.5 -6.6
 4.4 -5.5 6.6
 7.7 -8.8 9.9
 -7.7 8.8 -9.9
```

Zadatak 1.24 Na standardnom 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 1.19.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite elemente niza, nulu za kraj:
 1 2 3 0
 Trazena matrica je:
 1 2 3
 2 3 1
 3 1 2
```

Primer 2

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

Zadatak 1.25 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
```
3 Brazil
6 Nemacka
2 Kamerun
1 Brazil
2 Engleska
4 Engleska
5 Brazil
```

INTERAKCIJA SA PROGRAMOM:
```
Petru ukupno nedostaje 7 slicica.
Reprezentacija za koju je sakupio
najmanji broj slicica je Brazil.
```

*** Zadatak 1.26** 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

TEMENA.TXT
```
-1 -1
1 -1
1 1
-1 1
```

INTERAKCIJA SA PROGRAMOM:
```
U datoteci su zadata temena
cetvorougla.
Obim je 8.
Povrsina je 4.
```

Primer 2

TEMENA.TXT
```
-1.75 -1.5
3 1.5
2.2 3.1
-2 4
-4.1 1
```

INTERAKCIJA SA PROGRAMOM:
```
U datoteci su zadata temena
petougla.
Obim je 18.80.
Povrsina je 22.59.
```

1.4 Pokazivači na funkcije

Zadatak 1.27 Napisati program koji tabelarno štampa vrednosti proizvoljne realne funkcije sa jednim realnim argumentom, odnosno izračunava i ispisuje vrednosti date funkcije u n ekvidistantnih tačaka na intervalu $[a, b]$. Realni

brojevi a i b ($a < b$), kao i ceo broj n ($n \geq 2$), učitavaju se 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

```
POKRETANJE: ./a.out sin

INTERAKCIJA SA PROGRAMOM:
 Unesite krajeve intervala:
 -0.5 1
 Koliko tacaka ima na ekvidistantnoj
 mrezi (ukljucujuci krajeve intervala)?
 4
  x sin(x)
 -----------------------
 | -0.50000 | -0.47943 |
 | 0.00000 | 0.00000 |
 | 0.50000 | 0.47943 |
 | 1.00000 | 0.84147 |
 -----------------------
```

Primer 2

```
POKRETANJE: ./a.out cos

INTERAKCIJA SA PROGRAMOM:
 Unesite krajeve intervala:
 0 2
 Koliko tacaka ima na ekvidistantnoj
 mrezi (ukljucujuci krajeve intervala)?
 4
  x cos(x)
 -----------------------
 | 0.00000 | 1.00000 |
 | 0.66667 | 0.78589 |
 | 1.33333 | 0.23524 |
 | 2.00000 | -0.41615 |
 -----------------------
```

Zadatak 1.28 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:

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

Sa standardnog ulaza uneti ime funkcije i vrednosti n i a .

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

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

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
 Vrednost integrala je 0.645931.
```

Primer 1

```
Interakcija sa programom:
 Unesite ime funkcije, n, a i b:
 sin 10000 -5.2 2.1
 Vrednost integrala je 0.973993.
```

Zadatak 1.30 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 `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
 Vrednost integrala je 1.530295.
```

Primer 2

```
Interakcija sa programom:
 Unesite ime funkcije, n, a i b:
 tan 5000 -4.1 -2.3
 Vrednost integrala je -0.147640.
```

1.5 Rešenja

Rešenje 1.1

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

#define MAX 100

/* Funkcija obrce elemente niza koriscenjem indeksne 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;
  }
```

```c
}

/* Funkcija obrce elemente niza koriscenjem pokazivacke sintakse */
void obrni_niz_v2(int *a, int n)
{
  /* Pokazivaci na elemente niza */
  int *prvi, *poslednji;

  /* Vrsi se obrtanje niza */
  for (prvi = a, poslednji = a + n - 1; prvi < poslednji;) {
    int t = *prvi;

    /* Na adresu na koju pokazuje pokazivac prvi postavlja se
       vrednost koja se nalazi na adresi na koju pokazuje pokazivac
       poslednji. Nakon toga se pokazivac prvi uvecava za jedan sto
       za posledicu ima da prvi pokazuje na sledeci element u nizu */
    *prvi++ = *poslednji;

    /* Vrednost promenljive t se postavlja na adresu na koju
       pokazuje pokazivac poslednji. Ovaj pokazivac se zatim
       umanjuje za jedan, cime pokazivac poslednji pokazuje na
       element koji mu prethodi u nizu */
    *poslednji-- = t;
  }

  /*****************************************************************
    Drugi nacin za obrtanje niza

    for (prvi = a, poslednji = a + n - 1; prvi < poslednji;
                                          prvi++, poslednji--) {
      int t = *prvi;
      *prvi = *poslednji;
      *poslednji = t;
    }
   *****************************************************************/
}

int main()
{
  /* Deklarise se niz od najvise MAX elemenata */
  int a[MAX];

  /* Broj elemenata niza a */
  int n;

  /* Pokazivac na elemente niza */
  int *p;

  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);

  /* Proverava se da li je doslo do prekoracenja ogranicenja
```

```
68      dimenzije */
  if (n <= 0 || n > MAX) {
70    fprintf(stderr, "Greska: Neodgovarajuca dimenzija niza.\n");
    exit(EXIT_FAILURE);
72  }

74  printf("Unesite elemente niza:\n");
  for (p = a; p - a < n; p++)
76    scanf("%d", p);

78  obrni_niz_v1(a, n);

80  printf("Nakon obrtanja elemenata, niz je:\n");
  for (p = a; p - a < n; p++)
82    printf("%d ", *p);
  printf("\n");
84
  obrni_niz_v2(a, n);
86
  printf("Nakon ponovnog obrtanja elemenata, niz je:\n");
88  for (p = a; p - a < n; p++)
    printf("%d ", *p);
90  printf("\n");

92  exit(EXIT_SUCCESS);
}
```

Rešenje 1.2

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

4 #define MAX 100

6 /* Funkcija izracunava zbir elemenata niza */
double zbir(double *a, int n)
8 {
  double s = 0;
10  int i;

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

14  return s;
}
16
/* Funkcija izracunava proizvod elemenata niza */
18 double proizvod(double *a, int n)
{
20  double p = 1;

22  for (; n; n--)
```

```
    p *= (*(a + n - 1));

  return p;
}

/* Funkcija odredjuje minimalni element niza */
double min(double *a, int n)
{
  /* Na pocetku, minimalni element je prvi element */
  double min = *a;
  int i;

  /* Ispituje se da li se medju ostalim elementima niza nalazi
     minimalni */
  for (i = 1; i < n; i++)
    if (*(a + i) < min)
      min = *(a + i);

  return min;
}

/* Funkcija odredjuje maksimalni element niza */
double max(double *a, int n)
{
  /* Na pocetku, maksimalni element je prvi element */
  double max = *a;

  /* Ispituje se da li se medju ostalim elementima niza nalazi
     maksimalni */
  for (a++, n--; n > 0; a++, n--)
    if (*a > max)
      max = *a;

  return max;
}

int main()
{
  double a[MAX];
  int n, i;

  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);

  /* Proverava se da li je doslo do prekoracenja ogranicenja
     dimenzije */
  if (n <= 0 || n > MAX) {
    fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
    exit(EXIT_FAILURE);
  }

  printf("Unesite elemente niza:\n");
```

```
  for (i = 0; i < n; i++)
    scanf("%lf", a + i);

  /* Vrsi se testiranje definisanih funkcija */
  printf("Zbir elemenata niza je %5.3f.\n", zbir(a, n));
  printf("Proizvod elemenata niza je %5.3f.\n", proizvod(a, n));
  printf("Minimalni element niza je %5.3f.\n", min(a, n));
  printf("Maksimalni element niza je %5.3f.\n", max(a, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.3

```
#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 */
void povecaj_smanji(int *a, int n)
{
  int *prvi = a;
  int *poslednji = a + n - 1;

  while (prvi < poslednji) {

    /* Uvecava se element na koji pokazuje pokazivac prvi */
    (*prvi)++;

    /* Pokazivac prvi se pomera na sledeci element */
    prvi++;

    /* Smanjuje se vrednost elementa na koji pokazuje pokazivac
       poslednji */
    (*poslednji)--;

    /* Pokazivac poslednji se pomera na prethodni element */
    poslednji--;
  }

  /*****************************************************************
   Drugi nacin:
   while (prvi < poslednji) {
     (*prvi++)++;
     (*poslednji--)--;
   }
  *****************************************************************/
```

```
}

int main()
{
  int a[MAX];
  int n;
  int *p;

  printf("Unesite dimenziju niza: ");
  scanf("%d", &n);

  /* Proverava se da li je doslo do prekoracenja ogranicenja
     dimenzije */
  if (n <= 0 || n > MAX) {
    fprintf(stderr, "Greska: Neodgovarajuca dimenzija niza.\n");
    exit(EXIT_FAILURE);
  }

  printf("Unesite elemente niza:\n");
  for (p = a; p - a < n; p++)
    scanf("%d", p);

  povecaj_smanji(a, n);

  printf("Transformisan niz je:\n");
  for (p = a; p - a < n; p++)
    printf("%d ", *p);
  printf("\n");

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.4

```
#include <stdio.h>

int main(int argc, char *argv[])
{
  int i;
  char tip_ispisa;

  printf("Broj argumenata komandne linije je %d.\n", argc);
  printf("Kako zelite da ispisete argumente? Koriscenjem"
         " indeksne ili pokazivacke sintakse (I ili P)? ");
  scanf("%c", &tip_ispisa);

  printf("Argumenti komandne linije su:\n");
  if (tip_ispisa == 'I') {
    /* Ispisuju se argumenti komandne linije koriscenjem indeksne
       sintakse */
    for (i = 0; i < argc; i++)
```

```
      printf("%d %s\n", i, argv[i]);
  } else if (tip_ispisa == 'P') {
    /* Ispisuju se argumenti komandne linije koriscenjem
       pokazivacke sintakse */
    i = argc;
    for (; argc > 0; argc--)
      printf("%d %s\n", i - argc, *argv++);

    /* Nakon ove petlje argc je jednako nuli, a argv pokazuje na
       polje u memoriji koje se nalazi iza poslednjeg argumenta
       komandne linije. Kako je u promenljivoj i sacuvana vrednost
       broja argumenta komandne linije to sada moze ponovo da se
       postavi argv da pokazuje na nulti argument komandne linije */
    argv = argv - i;
    argc = i;
  }

  printf("Pocetna slova argumenata komandne linije:\n");
  if (tip_ispisa == 'I') {
    /* koristeci indeksnu sintaksu */
    for (i = 0; i < argc; i++)
      printf("%c ", argv[i][0]);
    printf("\n");
  } else if (tip_ispisa == 'P') {
    /* koristeci pokazivacku sintaksu */
    for (i = 0; i < argc; i++)
      printf("%c ", **argv++);
    printf("\n");
  }

  return 0;
}
```

Rešenje 1.5

```
#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;
  return 1;
}
```

```c
int main(int argc, char **argv)
{
  int i, n = 0;

  /* 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;
}
```

Rešenje 1.6

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

#define MAX_KARAKTERA 100

/* Implementacija funkcije strlen() iz standardne biblioteke */
int duzina(char *s)
{
  int i;
  for (i = 0; *(s + i); i++);
  return i;
}

int main(int argc, char **argv)
{
  char rec[MAX_KARAKTERA + 1];
  int br = 0, n;
  FILE *in;

  /* 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",
            argv[0]);
    exit(EXIT_FAILURE);
  }

  /* Otvara se datoteka sa imenom koje se zadaje kao prvi argument
     komandne linije. */
  in = fopen(*(argv + 1), "r");
  if (in == NULL) {
```

```
    fprintf(stderr, "Greska: ");
    fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
    exit(EXIT_FAILURE);
  }

  n = atoi(*(argv + 2));

  /* Broje se reci cija je duzina jednaka broju zadatom drugim
     argumentom komandne linije */
  while (fscanf(in, "%s", rec) != EOF)
    if (duzina(rec) == n)
      br++;

  printf("Broj reci ciji je broj karaktera %d je %d.\n", n, br);

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

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.7

```
#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);
}

/* Implementacija funkcije strcmp() iz standardne biblioteke */
int poredjenje_niski(char *s, char *t)
{
  int i;
  for (i = 0; *(s + i) == *(t + i); i++)
    if (*(s + i) == '\0')
      return 0;
  return *(s + i) - *(t + i);
}

/* Implementacija funkcije strlen() iz standardne biblioteke */
int duzina_niske(char *s)
{
  int i;
  for (i = 0; *(s + i); i++);
```

```
29    return i;
}

/* Funkcija ispituje da li je niska zadata drugim argumentom
33     funkcije sufiks niske zadate prvi argumentom funkcije */
int sufiks_niske(char *niska, char *sufiks)
35 {
    int duzina_sufiksa = duzina_niske(sufiks);
37    int duzina_niske_pom = duzina_niske(niska);
    if (duzina_sufiksa <= duzina_niske_pom &&
39        poredjenje_niski(niska + duzina_niske_pom -
                           duzina_sufiksa, sufiks) == 0)
41      return 1;
    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)
{
49    int i;
    int duzina_prefiksa = duzina_niske(prefiks);
51    int duzina_niske_pom = duzina_niske(niska);
    if (duzina_prefiksa <= duzina_niske_pom) {
53      for (i = 0; i < duzina_prefiksa; i++)
        if (*(prefiks + i) != *(niska + i))
55        return 0;
      return 1;
57    } else
      return 0;
59 }

61 int main(int argc, char **argv)
{
63    /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
       greska */
65    if (argc < 4) {
      fprintf(stderr, "Greska: ");
67      fprintf(stderr,
              "Nedovoljan broj argumenata komandne linije.\n");
69      fprintf(stderr, "Program se poziva sa\n");
      fprintf(stderr, "%s ime_dat suf/pref -s/-p\n", argv[0]);
71      exit(EXIT_FAILURE);
    }
73
    FILE *in;
75    int br = 0;
    char rec[MAX_KARAKTERA + 1];
77
    in = fopen(*(argv + 1), "r");
79    if (in == NULL) {
      fprintf(stderr, "Greska: ");
```

```
    fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n", argv[1]);
    exit(EXIT_FAILURE);
  }

  /* Proverava se opcija kojom je pozvan program, a zatim se
     ucitavaju reci iz datoteke i broji se koliko njih zadovoljava
     trazeni uslov */
  if (!(poredjenje_niski(*(argv + 3), "-s"))) {
    while (fscanf(in, "%s", rec) != EOF)
      br += sufiks_niske(rec, *(argv + 2));
    printf("Broj reci koje se zavrsavaju na %s je %d.\n",
           *(argv + 2), br);
  } else if (!(poredjenje_niski(*(argv + 3), "-p"))) {
    while (fscanf(in, "%s", rec) != EOF)
      br += prefiks_niske(rec, *(argv + 2));
    printf("Broj reci koje pocinju na %s je %d.\n", *(argv + 2),
           br);
  }

  fclose(in);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.8

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

#define MAX 100

/* 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;
}

/* Funkcija izracunava euklidsku normu matrice */
double euklidska_norma(int M[][MAX], int n)
{
  double norma = 0.0;
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      norma += M[i][j] * M[i][j];

```

```c
  return sqrt(norma);
}

/* Funkcija izracunava gornju vandijagonalnu normu matrice */
int gornja_vandijagonalna_norma(int M[][MAX], int n)
{
  int norma = 0;
  int i, j;

  for (i = 0; i < n; i++) {
    for (j = i + 1; j < n; j++)
      norma += abs(M[i][j]);
  }

  return norma;
}

int main()
{
  int A[MAX][MAX];
  int i, j, n;

  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);

  /* 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++)
      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.
   Kod ovako definisane matrice, elementi su uzastopno smesteni u
   memoriju, kao na traci. To znaci da su svi elementi prve vrste
   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++)
       printf("%d ", *(*(A+i)+j));
     printf("\n");
   }
   ****************************************************************/

  /* Ispisuje se rezultat na standardni izlaz */
  int tr = trag(A, n);
  printf("Trag matrice je %d.\n", tr);

  printf("Euklidska norma matrice je %.2f.\n",
         euklidska_norma(A, n));
  printf("Vandijagonalna norma matrice je = %d.\n",
         gornja_vandijagonalna_norma(A, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.9

```
#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++)
      scanf("%d", &m[i][j]);
}

/* 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++) {
    for (j = 0; j < n; j++)
      printf("%d ", m[i][j]);
    printf("\n");
  }
}

/* Funkcija proverava da li su zadate kvadratne matrice a i b
```

```c
   dimenzije n jednake */
int jednake_matrice(int a[][MAX], int b[][MAX], int n)
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (a[i][j] != b[i][j])
        return 0;

  /* Prosla je provera jednakosti za sve parove elemenata koji su
     na istim pozicijama. To znaci da su matrice jednake */
  return 1;
}

/* Funkcija izracunava zbir dve kvadratne matice */
void saberi(int a[][MAX], int b[][MAX], int c[][MAX], int n)
{
  int i, j;

  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;

  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];
    }
}

int main()
{
  /* 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 */
  int n;

  printf("Unesite broj vrsta matrica:\n");
  scanf("%d", &n);
```

```
83
   /* Proverava se da li je doslo do prekoracenja */
85   if (n > MAX || n <= 0) {
       fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
87     fprintf(stderr, "matrica.\n");
       exit(EXIT_FAILURE);
89   }

91   printf("Unesite elemente prve matrice po vrstama:\n");
   ucitaj_matricu(a, n);
93   printf("Unesite elemente druge matrice po vrstama:\n");
   ucitaj_matricu(b, n);
95
   /* Izracunava se zbir i proizvod matrica */
97   saberi(a, b, zbir, n);
   pomnozi(a, b, proizvod, n);
99
   /* Ispisuje se rezultat */
101  if (jednake_matrice(a, b, n) == 1)
       printf("Matrice su jednake.\n");
103  else
       printf("Matrice nisu jednake.\n");
105
   printf("Zbir matrica je:\n");
107  ispisi_matricu(zbir, n);

109  printf("Proizvod matrica je:\n");
   ispisi_matricu(proizvod, n);
111
   exit(EXIT_SUCCESS);
113 }
```

Rešenje 1.10

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

4 #define MAX 64

6 /* Funkcija proverava da li je relacija refleksivna. Relacija je
     refleksivna ako je svaki element u relaciji sa sobom, odnosno
8    ako se u matrici relacije na glavnoj dijagonali nalaze jedinice */
  int refleksivnost(int m[][MAX], int n)
10 {
   int i;
12
   for (i = 0; i < n; i++) {
14   if (m[i][i] != 1)
       return 0;
16 }
```

```
  return 1;
}

/* Funkcija prepisuje sadrzaj matrice original u matricu kopija */
void kopiraj_matricu(int original[][MAX], int n, int kopija[][MAX])
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      kopija[i][j] = original[i][j];
}

/* Funkcija odredjuje refleksivno zatvorenje zadate relacije. Ono
   je odredjeno matricom koja sadrzi sve elemente polazne matrice
   dopunjene jedinicama na glavnoj dijagonali */
void ref_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
{
  int i;

  /* Kopiraju se vrednosti elemenata pocetne matrice */
  kopiraj_matricu(m, n, zatvorenje);

  /* Na glavnoj dijagonali se postavljaju jedinice */
  for (i = 0; i < n; i++)
    zatvorenje[i][i] = 1;
}

/* Funkcija proverava da li je relacija simetricna. Relacija je
   simetricna ako za svaki par elemenata vazi: ako je element i u
   relaciji sa elementom j, onda je i element j u relaciji sa
   elementom i. Ovakve matrice su simetricne u odnosu na glavnu
   dijagonalu */
int simetricnost(int m[][MAX], int n)
{
  int i, j;

  /* Obilaze se elementi ispod glavne dijagonale matrice i
     uporedjuju se sa njima simetricnim elementima */
  for (i = 0; i < n; i++)
    for (j = 0; j < i; j++)
      if (m[i][j] != m[j][i])
        return 0;

  return 1;
}

/* 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 */
void sim_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
```

```c
{
  int i, j;

  /* Kopiraju se vrednosti elemenata pocetne matrice */
  kopiraj_matricu(m, n, zatvorenje);

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (zatvorenje[i][j] == 1)
        zatvorenje[j][i] = 1;
}

/* 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)
{
  int i, j, k;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      /* Ispituje se da li postoji element koji narusava *
         tranzitivnost */
      for (k = 0; k < n; k++)
        if (m[i][k] == 1 && m[k][j] == 1 && m[i][j] == 0)
          return 0;

  return 1;
}


/* Funkcija odredjuje refleksivno-tranzitivno zatvorenje zadate
   relacije koriscenjem Varsalovog algoritma */
void ref_tran_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
{
  int i, j, k;

  /* Odredjuje se refleksivno zatvorenje matrice  */
  ref_zatvorenje(m, n, zatvorenje);

  /* Primenom Varsalovog algoritma odredjuje se tranzitivno
     zatvorenje matrice */
  for (k = 0; k < n; k++)
    for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
        if ((zatvorenje[i][k] == 1) && (zatvorenje[k][j] == 1)
            && (zatvorenje[i][j] == 0))
          zatvorenje[i][j] = 1;
}

/* Funkcija ispisuje elemente matrice */
```

```c
void pisi_matricu(int m[][MAX], int n)
{
  int i, j;

  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
      printf("%d ", m[i][j]);
    printf("\n");
  }
}

int main(int argc, char *argv[])
{
  FILE *ulaz;
  int m[MAX][MAX];
  int pomocna[MAX][MAX];
  int n, i, j;

  /* Proverava se da li korisnik nije uneo trazene argumente */
  if (argc < 2) {
    printf("Greska: ");
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_dat.\n", argv[0]);
    exit(EXIT_FAILURE);
  }

  /* 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);
  }

  /* Ucitava se dimenzija matrice */
  fscanf(ulaz, "%d", &n);

  /* Proverava se da li je doslo do prekoracenja dimenzije */
  if (n > MAX || n <= 0) {
    fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
    fprintf(stderr, "matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitava se element po element matrice */
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      fscanf(ulaz, "%d", &m[i][j]);

  /* Ispisuje se rezultat */
  printf("Relacija %s refleksivna.\n",
         refleksivnost(m, n) == 1 ? "jeste" : "nije");
```

```
174
   printf("Relacija %s simetricna.\n",
176           simetricnost(m, n) == 1 ? "jeste" : "nije");

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

196   exit(EXIT_SUCCESS);
}
```

Rešenje 1.11

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

4 #define MAX 32

6 /* Funkcija izracunava najveci element na sporednoj dijagonali. Za
     elemente sporedne dijagonale vazi da je zbir indeksa vrste i
8    indeksa kolone jednak n-1 */
int max_sporedna_dijagonala(int m[][MAX], int n)
10 {
   int i;
12   int max_na_sporednoj_dijagonali = m[0][n - 1];

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

18   return max_na_sporednoj_dijagonali;
}
20
/* Funkcija izracunava indeks kolone najmanjeg elementa */
22 int indeks_min(int m[][MAX], int n)
{
24   int i, j;
```

```
  int min = m[0][0], indeks_kolone = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (m[i][j] < min) {
        min = m[i][j];
        indeks_kolone = j;
      }

  return indeks_kolone;
}

/* Funkcija izracunava indeks vrste najveceg elementa */
int indeks_max(int m[][MAX], int n)
{
  int i, j;
  int max = m[0][0], indeks_vrste = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (m[i][j] > max) {
        max = m[i][j];
        indeks_vrste = i;
      }
  return indeks_vrste;
}

/* Funkcija izracunava broj negativnih elemenata matrice */
int broj_negativnih(int m[][MAX], int n)
{
  int i, j;
  int broj_negativnih = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (m[i][j] < 0)
        broj_negativnih++;

  return broj_negativnih;
}

int main(int argc, char *argv[])
{
  int m[MAX][MAX];
  int n;
  int i, j;

  /* Proverava se broj argumenata komandne linije */
  if (argc < 2) {
    printf("Greska: ");
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s br_vrsta_mat.\n", argv[0]);
```

```
    exit(EXIT_FAILURE);
  }

  /* Ucitava se broj vrsta matrice */
  n = atoi(argv[1]);

  if (n > MAX || n <= 0) {
    fprintf(stderr, "Greska: Neodgovarajuci broj ");
    fprintf(stderr, "vrsta matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitava se matrica */
  printf("Unesite elemente matrice dimenzije %dx%d:\n", n, n);
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &m[i][j]);

  /* Ispisuju se rezultati izracunavanja */
  printf("Najveci element sporedne dijagonale je %d.\n",
         max_sporedna_dijagonala(m, n));

  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));

  printf("Broj negativnih elemenata matrice je %d.\n",
         broj_negativnih(m, n));

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.12

```
#include <stdio.h>
#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++)
      scanf("%d", &m[i][j]);
}
```

```c

/* 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++) {
    for (j = 0; j < n; j++)
      printf("%d ", m[i][j]);
    printf("\n");
  }
}

/* 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)
{
  int i, j, k;
  int proizvod;

  /* Ispituje se uslov normiranosti */
  for (i = 0; i < n; i++) {
    proizvod = 0;
    for (j = 0; j < n; j++)
      proizvod += m[i][j] * m[i][j];
    if (proizvod != 1)
      return 0;
  }

  /* 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;
    }
  }

  /* Ako su oba uslova ispunjena, matrica je ortonormirana */
  return 1;
}

int main()
{
  int A[MAX][MAX];
  int n;
```

```
  printf("Unesite broj vrsta matrice: ");
  scanf("%d", &n);

  if (n > MAX || n <= 0) {
    fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
    fprintf(stderr, "matrice.\n");
    exit(EXIT_FAILURE);
  }

  printf("Unesite elemente matrice po vrstama:\n");
  ucitaj_matricu(A, n);
  printf("Matrica %s ortonormirana.\n",
         ortonormirana(A, n) ? "je" : "nije");

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.13

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

#define MAX_V 10
#define MAX_K 10

/* Funkcija proverava da li su ispisani svi elementi iz matrice,
   odnosno da li se narusio prirodan poredak medju granicama */
int kraj_ispisa(int vrh, int dno, int levo, int desno)
{
  return !(vrh <= dno && levo <= desno);
}

/* Funkcija spiralno ispisuje elemente matrice */
void ispisi_matricu_spiralno(int a[][MAX_K], int n, int m)
{
  int i, j, vrh, dno, levo, desno;

  vrh = levo = 0;
  dno = n - 1;
  desno = m - 1;

  while (!kraj_ispisa(vrh, dno, levo, desno)) {
    for (j = levo; j <= desno; j++)
      printf("%d ", a[vrh][j]);

    /* Spusta se prvi red za naredni krug ispisa */
    vrh++;

    if (kraj_ispisa(vrh, dno, levo, desno))
      break;
```

```
32
    for (i = vrh; i <= dno; i++)
34    printf("%d ", a[i][desno]);

36  /* Pomera se desna kolona za naredni krug ispisa blize levom
       kraju */
38  desno--;

40  if (kraj_ispisa(vrh, dno, levo, desno))
      break;
42
    /* Ispisuje se donja vrsta */
44  for (j = desno; j >= levo; j--)
      printf("%d ", a[dno][j]);
46
    /* Podize se donja vrsta za naredni krug ispisa */
48  dno--;

50  if (kraj_ispisa(vrh, dno, levo, desno))
      break;
52
    /* Ispisuje se prva kolona */
54  for (i = dno; i >= vrh; i--)
      printf("%d ", a[i][levo]);
56
    /* Priprema se leva kolona za naredni krug ispisa */
58  levo++;
  }
60  putchar('\n');
}
62
/* Funkcija ucitava matricu */
64 void ucitaj_matricu(int a[][MAX_K], int n, int m)
{
66  int i, j;

68  for (i = 0; i < n; i++)
      for (j = 0; j < m; j++)
70      scanf("%d", &a[i][j]);
}
72
int main()
74 {
  int a[MAX_V][MAX_K];
76  int m, n;

78  printf("Unesite broj vrsta i broj kolona:\n");
  scanf("%d %d", &n, &m);
80
  if (n > MAX_V || n <= 0 || m > MAX_K || m <= 0) {
82    fprintf(stderr, "Greska: neodgovarajuce dimenzije ");
    fprintf(stderr, "matrice.\n");
```

```
    exit(EXIT_FAILURE);
  }

  printf("Unesite elemente matrice po vrstama:\n");
  ucitaj_matricu(a, n, m);

  printf("Spiralno ispisana matrica: ");
  ispisi_matricu_spiralno(a, n, m);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.15

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

int main()
{
  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) {
    fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
    exit(EXIT_FAILURE);
  }

  printf("Unesite elemente niza: ");
  for (i = 0; i < n; i++)
    scanf("%d", &p[i]);

  printf("Niz u obrnutom poretku je: ");
  for (i = n - 1; i >= 0; i--)
    printf("%d ", p[i]);
  printf("\n");

  /* Oslobadja se prostor rezervisan funkcijom malloc() */
  free(p);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.16

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

#define KORAK 10

int main()
{
  /* Adresa prvog alociranog bajta */
  int *a = NULL;

  /* Velicina alocirane memorije */
  int alocirano;

  /* Broj elemenata niza */
  int n;

  /* Broj koji se ucitava sa ulaza */
  int x;
  int i;
  int *b = NULL;
  char realokacija;

  /* Inicijalizacija */
  alocirano = n = 0;

  printf("Unesite zeljeni nacin realokacije (M ili R):\n");
  scanf("%c", &realokacija);

  printf("Unesite brojeve, nulu za kraj:\n");
  scanf("%d", &x);

  while (x != 0) {
    if (n == alocirano) {
      alocirano = alocirano + KORAK;

      if (realokacija == 'M') {
        /* Vrsi se realokacija memorije sa novom velicinom */
        b = (int *) malloc(alocirano * sizeof(int));

        if (b == NULL) {
          fprintf(stderr,
                  "Greska: Neuspesna alokacija memorije.\n");
          free(a);
          exit(EXIT_FAILURE);
        }

        /* Svih n elemenata koji pocinju na adresi a prepisuju se
           na novu aderesu b */
        for (i = 0; i < n; i++)
          b[i] = a[i];
```

```
52          free(a);

54          /* Promenljivoj a dodeljuje se adresa pocetka novog, veceg
               bloka cija je adresa prilikom alokacije zapamcena u
56             promenljivoj b */
            a = b;
58        } else if (realokacija == 'R') {

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

78  /* Oslobadja se dinamicki alocirana memorija */
    free(a);
80
    exit(EXIT_SUCCESS);
82 }
```

Rešenje 1.17

```
#include <stdio.h>
2 #include <stdlib.h>
  #include <string.h>
4
  #define MAX 1000
6
  /* Funkcija dinamicki kreira niz karaktera u koji smesta rezultat
8    nadovezivanja niski. Adresa kreiranog niza se vraca kao povratna
     vrednost. */
10 char *nadovezi(char *s, char *t)
  {
12   char *p = (char *) malloc((strlen(s) + strlen(t) + 1)
                              * sizeof(char));
14
     /* Proverava se da li je memorija uspesno alocirana */
16   if (p == NULL) {
```

```
    fprintf(stderr, "Greska: Neuspesna alokacija memorije.\n");
    exit(EXIT_FAILURE);
  }

  /* Kopiraju se i nadovezuju niske karaktera */
  strcpy(p, s);
  strcat(p, t);

  return p;
}

int main()
{
  char *s = NULL;
  char s1[MAX], s2[MAX];

  printf("Unesite dve niske karaktera:\n");
  scanf("%s", s1);
  scanf("%s", s2);

  /* Poziva se funkcija koja nadovezuje niske */
  s = nadovezi(s1, s2);

  /* Prikazuje se rezultat */
  printf("Nadovezane niske: %s\n", s);

  /* Oslobadja se memorija alocirana u funkciji nadovezi() */
  free(s);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.18

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

int main()
{
  int i, j;

  /* Pokazivac na niz vrsta matrice realnih brojeva */
  double **A = NULL;

  /* Broj vrsta i broj kolona */
  int n = 0, m = 0;

  /* Trag matice */
  double trag = 0;

```

```c
  printf("Unesite broj vrsta i broj kolona:\n ");
  scanf("%d%d", &n, &m);

  /* Dinamicki se rezervise prostor za niz vrsta matrice */
  A = (double **) malloc(sizeof(double *) * n);

  /* Proverava se da li je uspelo rezervisanje memorije */
  if (A == NULL) {
    fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
    exit(EXIT_FAILURE);
  }

  /* Dinamicki se rezervise prostor za elemente u vrstama */
  for (i = 0; i < n; i++) {
    A[i] = (double *) malloc(sizeof(double) * m);

    /* Ukoliko je alokacija neuspesna, pre zavrsetka programa
       potrebno je osloboditi svih i-1 prethodno alociranih vrsta,
       i alociran niz pokazivaca */
    if (A[i] == NULL) {
      for (j = 0; j < i; j++)
        free(A[j]);
      free(A);
      exit(EXIT_FAILURE);
    }
  }

  printf("Unesite elemente matrice po vrstama:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%lf", &A[i][j]);

  /* Izracunava se trag matrice, odnosno suma elemenata na glavnoj
     dijagonali */
  trag = 0.0;

  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 */
  for (j = 0; j < n; j++)
    free(A[j]);

  /* Oslobadja se memorija za niz pokazivaca na vrste */
  free(A);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.19

matrica.h

```
#ifndef _MATRICA_H_
#define _MATRICA_H_ 1

/* Funkcija dinamicki alocira memoriju za matricu dimenzije n x m */
int **alociraj_matricu(int n, int m);

/* Funkcija dinamicki alocira memoriju za kvadratnu matricu
   dimenzije n x n */
int **alociraj_kvadratnu_matricu(int n);

/* 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 */
void ucitaj_kvadratnu_matricu(int **matrica, int n);

/* Funkcija ispisuje matricu dimenzije n x m na standardni izlaz */
void ispisi_matricu(int **matrica, int n, int m);

/* Funkcija ispisuje kvadratnu matricu dimenzije n na standardni
   izlaz */
void ispisi_kvadratnu_matricu(int **matrica, int n);

/* Funkcija ucitava vec alociranu matricu dimenzije n x m iz
   datoteke f */
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);

/* 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 */
int upisi_kvadratnu_matricu_u_datoteku(int **matrica, int n,
                                       FILE * f);

#endif
```

matrica.c

```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)
    return NULL;

  /* Alocira se prostor za svaku vrstu matrice */
  for (i = 0; i < n; i++) {
    matrica[i] = (int *) malloc(m * sizeof(int));
    /* Ako alokacija nije prosla uspesno, oslobadjaju se svi
       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;
    }
  }
  return matrica;
}

int **alociraj_kvadratnu_matricu(int n)
{
  return alociraj_matricu(n, n);
}

int **dealociraj_matricu(int **matrica, int n)
{
  int i;
  /* 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);

  /* Matrica postaje prazna, tj. nealocirana */
  return NULL;
}
```

```c
void ucitaj_matricu(int **matrica, int n, int m)
{
  int i, j;
  /* Elementi matrice se ucitavaju po vrstama */
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%d", &matrica[i][j]);
}

void ucitaj_kvadratnu_matricu(int **matrica, int n)
{
  ucitaj_matricu(matrica, n, n);
}

void ispisi_matricu(int **matrica, int n, int m)
{
  int i, j;
  /* Ispis po vrstama */
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
      printf("%d ", matrica[i][j]);
    printf("\n");
  }
}

void ispisi_kvadratnu_matricu(int **matrica, int n)
{
  ispisi_matricu(matrica, n, n);
}

int ucitaj_matricu_iz_datoteke(int **matrica, int n, int m,
                               FILE * f)
{
  int i, j;
  /* Elementi matrice se ucitavaju po vrstama */
  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      /* Ako je nemoguce ucitati sledeci element, povratna vrednost
         funkcije je 1, kao indikator neuspesnog ucitavanja */
      if (fscanf(f, "%d", &matrica[i][j]) != 1)
        return 1;

  /* Uspesno ucitana matrica */
  return 0;
}

int ucitaj_kvadratnu_matricu_iz_datoteke(int **matrica, int n,
                                         FILE * f)
{
  return ucitaj_matricu_iz_datoteke(matrica, n, n, f);
}
```

```c
int upisi_matricu_u_datoteku(int **matrica, int n, int m, FILE * f)
{
  int i, j;
  /* Ispis po vrstama */
  for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++)
      /* Ako je nemoguce ispisati sledeci element, povratna
         vrednost funkcije je 1, kao indikator neuspesnog ispisa */
      if (fprintf(f, "%d ", matrica[i][j]) <= 0)
        return 1;
    fprintf(f, "\n");
  }

  /* Uspesno upisana matrica */
  return 0;
}

int upisi_kvadratnu_matricu_u_datoteku(int **matrica, int n,
                                       FILE * f)
{
  return upisi_matricu_u_datoteku(matrica, n, n, f);
}
```

main_a.c

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

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);
  printf("Unesite broj kolona matrice: ");
  scanf("%d", &m);

  /* Provera dimenzija matrice */
  if (n <= 0 || m <= 0) {
    fprintf(stderr,
            "Greska: Broj vrsta i broj kolona ne mogu biti negativni
    brojevi.\n");
    exit(EXIT_FAILURE);
  }

  /* Rezervise se memorijski prostor za matricu i proverava se da
     li je memorijski prostor uspesno rezervisan */
```

```
  matrica = alociraj_matricu(n, m);
  if (matrica == NULL) {
    fprintf(stderr, "Greska: Neuspesna alokacija matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitava se matrica sa standardnog ulaza */
  printf("Unesite elemente matrice po vrstama:\n");
  ucitaj_matricu(matrica, n, m);

  /* Otvara se datoteka za upis matrice */
  if ((f = fopen("matrica.txt", "w")) == NULL) {
    fprintf(stderr, "Greska: Neuspesno otvaranje datoteke.\n");
    matrica = dealociraj_matricu(matrica, n);
    exit(EXIT_FAILURE);
  }

  /* Upis matrice u datoteku */
  if (upisi_matricu_u_datoteku(matrica, n, m, f) != 0) {
    fprintf(stderr,
            "Greska: Neuspesno upisivanje matrice u datoteku.\n");
    matrica = dealociraj_matricu(matrica, n);
    exit(EXIT_FAILURE);
  }

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

  /* Oslobadja se memorija koju je zauzimala matrica */
  matrica = dealociraj_matricu(matrica, n);

  exit(EXIT_SUCCESS);
}
```

main_b.c

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

int main(int argc, char **argv)
{
  int **matrica = NULL;
  int n;
  FILE *f;

  /* Provera argumenata komandne linije */
  if (argc != 2) {
    fprintf(stderr, "Greska: Koriscenje programa: %s datoteka\n",
            argv[0]);
    exit(EXIT_FAILURE);
```

```
  }

  /* Otvara se datoteka za citanje */
  if ((f = fopen(argv[1], "r")) == NULL) {
    fprintf(stderr, "Greska: Neuspesno otvaranje datoteke.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitava se dimenzija matrice */
  if (fscanf(f, "%d", &n) != 1) {
    fprintf(stderr, "Greska: Neispravan pocetak datoteke.\n");
    exit(EXIT_FAILURE);
  }

  /* Provera dimenzija matrice */
  if (n <= 0) {
    fprintf(stderr, "Greska: Neodgovarajca dimenzija matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Rezervise se memorijski prostor za matricu i vrsi se provera */
  matrica = alociraj_kvadratnu_matricu(n);
  if (matrica == NULL) {
    fprintf(stderr, "Greska: Neuspesna alokacija matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Ucitava se matrica iz datoteke */
  if (ucitaj_kvadratnu_matricu_iz_datoteke(matrica, n, f) != 0) {
    fprintf(stderr,
            "Greska: Neuspesno ucitavanje matrice iz datoteke.\n");
    matrica = dealociraj_matricu(matrica, n);
    exit(EXIT_FAILURE);
  }

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

  /* Ispis matrice na standardni izlaz */
  ispisi_kvadratnu_matricu(matrica, n);

  /* Oslobadja se memorija koju je zauzimala matrica */
  matrica = dealociraj_matricu(matrica, n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.20

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

```c
#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]);
    printf("\n");
  }
}

int main()
{
  int m, n;
  int **matrica = NULL;

  printf("Unesite broj vrsta i broj kolona:\n ");
  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);
  }

  printf("Unesite elemente matrice po vrstama:\n");
  ucitaj_matricu(matrica, n, m);

  printf("Elementi ispod glavne dijagonale matrice:\n");
  ispisi_elemente_ispod_dijagonale(matrica, n, m);

  /* Oslobadja se memorija */
  matrica = dealociraj_matricu(matrica, n);

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.22

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

/* Funkcija izvrsava trazene transformacije nad matricom */
void izmeni(float **a, int n)
```

```c
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (i < j)
        a[i][j] /= 2;
      else if (i > j)
        a[i][j] *= 2;
}

/* 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
   n-1 */
float zbir_ispod_sporedne_dijagonale(float **m, int n)
{
  int i, j;
  float zbir = 0;

  for (i = 0; i < n; i++)
    for (j = n - i; j < n; j++)
      if (i + j > n - 1)
        zbir += fabs(m[i][j]);

  return zbir;
}

/* Funkcija ucitava elemente kvadratne matrice dimenzije n x n iz
   zadate datoteke */
void ucitaj_matricu(FILE * ulaz, float **m, int n)
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      fscanf(ulaz, "%f", &m[i][j]);
}

/* Funkcija ispisuje elemente kvadratne matrice dimenzije n x n na
   standardni izlaz */
void ispisi_matricu(float **m, int n)
{
  int i, j;

  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
      printf("%.2f ", m[i][j]);
    printf("\n");
  }
}

```

```c
/* Funkcija alocira memoriju za kvadratnu matricu dimenzije n x n */
float **alociraj_memoriju(int n)
{
  int i, j;
  float **m;

  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) {
      fprintf(stderr, "Greska: Neupesna alokacija memorije.\n");
      for (j = 0; j < i; j++)
        free(m[i]);
      free(m);
      exit(EXIT_FAILURE);
    }
  }
  return m;
}

/* Funkcija oslobadja memoriju zauzetu kvadratnom matricom
   dimenzije n x n */
void oslobodi_memoriju(float **m, int n)
{
  int i;

  for (i = 0; i < n; i++)
    free(m[i]);
  free(m);
}

int main(int argc, char *argv[])
{
  FILE *ulaz;
  float **a;
  int n;

  /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
     greska */
  if (argc < 2) {
    printf("Greska: ");
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_dat.\n", argv[0]);
    exit(EXIT_FAILURE);
  }

```

```
  /* 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);
  }

  /* Cita se dimenzija matrice */
  fscanf(ulaz, "%d", &n);

  /* Rezervise se memorija */
  a = alociraj_memoriju(n);

  /* Ucitavaju se elementi matrice */
  ucitaj_matricu(ulaz, a, n);

  float zbir = zbir_ispod_sporedne_dijagonale(a, n);

  /* Poziva se funkcija za transformaciju matrice */
  izmeni(a, n);

  /* Ispisuju se rezultati */
  printf("Zbir apsolutnih vrednosti ispod sporedne dijagonale ");
  printf("je %.2f.\n", zbir);

  printf("Transformisana matrica je:\n");
  ispisi_matricu(a, n);

  /* Oslobadja se memorija */
  oslobodi_memoriju(a, n);

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

  exit(EXIT_SUCCESS);
}
```

Rešenje 1.27

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

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

double sqr(double a)
{
  return a * a;
}

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);

  /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
     greska */
  if (argc < 2) {
    fprintf(stderr, "Greska: ");
    fprintf(stderr,
            "Nedovoljan broj argumenata komandne linije.\n");
    fprintf(stderr,
            "Program se poziva sa %s ime_funkcije iz math.h.\n",
            argv[0]);
    exit(EXIT_FAILURE);
  }

  /* 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;
  else if (strcmp(ime_funkcije, "tan") == 0)
    fp = &tan;
  else if (strcmp(ime_funkcije, "atan") == 0)
```

```
    fp = &atan;
  else if (strcmp(ime_funkcije, "acos") == 0)
    fp = &acos;
  else if (strcmp(ime_funkcije, "asin") == 0)
    fp = &asin;
  else if (strcmp(ime_funkcije, "exp") == 0)
    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;
  else if (strcmp(ime_funkcije, "sqr") == 0)
    fp = &sqr;
  else {
    fprintf(stderr, "Greska");
    fprintf(stderr,
            "Program jos uvek ne podrzava trazenu funkciju!\n");
    exit(EXIT_FAILURE);
  }

  printf("Unesite krajeve intervala:\n");
  scanf("%lf %lf", &a, &b);

  printf("Koliko tacaka ima na ekvidistantnoj mrezi ");
  printf("(ukljucujuci krajeve intervala)?\n");
  scanf("%d", &n);

  /* Mreza mora da ukljucuje bar krajeve intervala, tako da se mora
     uneti broj veci od 2 */
  if (n < 2) {
    fprintf(stderr, "Greska: Broj tacaka mreze mora biti bar 2!\n");
    exit(EXIT_FAILURE);
  }

  /* Ispisuje se ime funkcije */
  printf("     x %10s(x)\n", ime_funkcije);

  /* Funkciji tabela() se prosledjuje funkcija koja je zadata kao
     argument komandne linije */
  tabela(a, b, n, fp);

  exit(EXIT_SUCCESS);
}
```