Univerzitet u Beogradu
Matematički fakultet

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

Programiranje 2
Zbirka zadataka sa rešenjima

Beograd, 2015.

Predgovor

U okviru kursa *Programiranje 2* na Matematičkom fakultetu vežbaju se zadaci koji imaju za cilj da studente nauče rekurzivnom pristupu rešavanju 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 kolokvijuma i ispita. Elektronska verzija zbirke, dostupna je u okviru strane kursa `www.programiranje2.matf.bg.ac.rs`, a tu je dostupan i radni repozitorijum elektronskih verzija rešenja zadataka.

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, pomenimo tu, pre svega, Milana Bankovića i doc dr Filipa Marića. Zbog toga smatramo da je naš osnovni doprinos što smo objedinili, precizno formulisali i rešili sve najvažnije zadatke koji su potrebni za uspešno savlađivanje koncepata koji se obrađuju u okviru kursa.

...

Autori

Sadržaj

Glava 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 programa:
 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 programa:
 Unesite dimenziju niza:  0
 Greska: neodgovarajuca dimenzija niza.
```

[Rešenje 1.1]

Zadatak 1.2 Dat je niz realnih brojeva dimenzije n.

(a) Napisati funkciju `zbir` koja izračunava zbir elemenata niza.

(b) Napisati funkciju `proizvod` koja izračunava proizvod elemenata niza.

(c) Napisati funkciju `min_element` koja izračunava najmanji elemenat niza.

(d) Napisati 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 programa:
 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 programa:
 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.
```

[Rešenje 1.2]

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

Primer 1

```
Interakcija programa:
 Unesite dimenziju niza:  5
 Unesite elemente niza:
 1 2 3 4 5
 Transformisan niz je:
 2 3 3 3 4
```

Primer 2

```
Interakcija programa:
 Unesite dimenziju niza:  4
 Unesite elemente niza:
 4 -3 2 -1
 Transformisan niz je:
 5 -2 1 -2
```

Primer 3

```
Interakcija programa:
 Unesite dimenziju niza:  0
 Greska: neodgovarajuca dimenzija niza.
```

Primer 4

```
Interakcija programa:
 Unesite dimenziju niza:  101
 Greska: neodgovarajuca dimenzija niza.
```

[Rešenje 1.3]

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

Primer 1

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

Interakcija programa:
 Broj prihvacenih 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 su:
 . p 2 t -
```

Primer 2

```
Poziv: ./a.out

Interakcija programa:
 Broj prihvacenih 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 su:
 .
```

[Rešenje 1.4]

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

```
Poziv: ./a.out a b 11 212

Interakcija programa:
 Broj argumenata komandne linije
 koji su palindromi je 4.
```

Primer 2

```
Poziv: ./a.out

Interakcija programa:
 Broj argumenata komandne linije koji
 koji su palindromi je 0.
```

[Rešenje 1.5]

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

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

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

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

Primer 2

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

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

INTERAKCIJA PROGRAMA:
 Greska: Nedovoljan broj argumenata
 komandne linije.
 Program se poziva sa
 ./a.out ime_dat br_karaktera.
```

Primer 3

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

DATOTEKA ULAZ.TXT NE POSTOJI

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

[Rešenje 1.6]

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 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

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

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

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

Primer 2

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

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

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

Primer 3

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

Datoteka ulaz.txt ne postoji

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

Primer 4

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

ulaz.txt
  Ovo je sadrzaj ulaza.

Interakcija programa:
  Greska: Nedovoljan broj argumenata
  komandne linije.
  Program se poziva sa
  ./a.out ime_dat suf/pref -s/-p.
```

[Rešenje 1.7]

1.2 Višedimenzioni nizovi

Zadatak 1.8 Data je kvadratna matrica dimenzije 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 dimanziju 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 programa:
  Unesite dimenziju matrice:  3
  Unesite elemente matrice, vrstu po vrstu:
  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 programa:
  Unesite dimenziju matrice:  0
  Greska: neodgovarajuca dimenzija matrice.
```

[Rešenje 1.8]

Zadatak 1.9 Date su dve kvadratne matrice istih dimenzija 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 dimanziju 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 programa:
 Unesite dimenziju matrica:  3
 Unesite elemente prve matrice, vrstu po vrstu:
 1 2 3
 1 2 3
 1 2 3
 Unesite elemente druge matrice, vrstu po vrstu:
 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
```

[Rešenje 1.9]

Zadatak 1.10 Relacija se može predstaviti kvadratnom matricom nula i jedinica na sledeći način: dva elementa i i j su u relaciji ukoliko se u preseku i-te vrste i j-te kolone matrice nalazi broj 1, a nisu 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 sadrži datu).

(e) Napisati funkciju koja određuje simetrično zatvorenje relacije (najmanju simetričnu relaciju koja sadrži datu).

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

Primer 1

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

ulaz.txt
 4
 1 0 0 0
 0 1 1 0
 0 0 1 0
 0 0 0 0

Interakcija programa:
 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
```

[Rešenje 1.10]

Zadatak 1.11 Data je kvadratna matrica dimenzije 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 čija se dimenzija n ($0 < n \leq 32$) zadaje kao argument komandne linije. Na standardni izlaz ispisati rezultat primene prethodno napisanih funkcija.

Primer 1

```
Poziv: ./a.out 3

Interakcija programa:
 Unesite elemente matrice dimenzije 3:
 1 2 3
 -4 -5 -6
 7 8 9
 Najveci element matrice na sporednoj dijagonali je 7.
 Indeks kolone koja sadrzi najmanji element matrice 2.
 Indeks vrste koja sadrzi najveci element matrice 2.
 Broj negativnih elemenata matrice je 3.
```

Primer 2

```
Poziv: ./a.out 4

Interakcija programa:
 Unesite elemente matrice dimenzije 4:
 -1 -2 -3 -4
 -5 -6 -7 -8
 -9 -10 -11 -12
 -13 -14 -15 -16
 Najveci element matrice na sporednoj dijagonali je -4.
 Indeks kolone koja sadrzi najmanji element matrice 3.
 Indeks vrste koja sadrzi najveci element matrice 0.
 Broj negativnih elemenata matrice je 16.
```

[Rešenje 1.11]

Zadatak 1.12 Napisati funkciju kojom se proverava da li je zadata kvadratna matrica dimenzije 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 dimenziju 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 programa:
 Unesite dimenziju matrice:  4
 Unesite elemente matrice, vrstu po vrstu:
 1 0 0 0
 0 1 0 0
 0 0 1 0
 0 0 0 1
 Matrica je ortonormirana.
```

Primer 2

```
Interakcija programa:
 Unesite dimenziju matrice:  3
 Unesite elemente matrice, vrstu po vrstu:
 1 2 3
 5 6 7
 1 4 2
 Matrica nije ortonormirana.
```

[Rešenje 1.12]

Zadatak 1.13 Data je matrica dimenzije $n \times m$.

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

(b) Napsiati funkciju koja na standardni izlaz spiralno ispisuje elemente matrice.

Napisati program koji testira napisane funkcije. Sa standardnog ulaza učitati dimenzije matrice n ($0 < n \leq 10$) i m ($0 < n \leq 10$), a zatim i elemente matrice (pozivom gore napisane funkcije). Na standardni izlaz spiralno ispisati elemente učitane matrice.

Primer 1

```
Interakcija programa:
 Unesite broj vrsta i broj kolona matrice:
 3 3
 Unesite elemente matrice, vrstu po vrstu:
 1 2 3
 4 5 6
 7 8 9
 Spiralno ispisana matrica:
 1 2 3 6 9 8 7 4 5
```

Primer 2

```
Interakcija programa:
 Unesite broj vrsta i broj kolona matrice:
 3 4
 Unesite elemente matrice, vrstu po vrstu:
 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
```

[Rešenje 1.13]

Zadatak 1.14 Napisati funkciju koja izračunava k-ti stepen kvadratne matrice dimenzije n ($0 < n \leq 32$). Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju 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 programa:
 Unesite dimenziju kvadratne matrice:  3
 Unesite elemente matrice, vrstu po vrstu:
 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 programa:
 Unesite dimenziju niza:  3
 Unesite elemente niza:
 1 -2 3
 Niz u obrnutom poretku je: 3 -2 1
```

Primer 2

```
Interakcija programa:
 Unesite dimenziju niza:  -1
 malloc(): neuspela alokacija memorije.
```

[Rešenje 1.15]

Zadatak 1.16 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.

Primer 1

```
Interakcija programa:
 Unesite brojeve, nulu za kraj:
 1 -2 3 -4 0
 Unesite elemente niza:
 1 -2 3 -4 0
 Niz u obrnutom poretku je: -4 3 -2 1
```

[Rešenje 1.16]

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 nisu duže od 1000 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 PROGRAMA:
 Unesite dve niske karaktera:
 Jedan Dva
 Nadovezane niske: JedanDva
```

[Rešenje 1.17]

Zadatak 1.18 Napisati program koji sa standardnog ulaza učitava matricu realnih brojeva. Prvo se učitavaju dimenzije matrice n i m (ne praviti nikakve pretpostavke o njihovoj veličini), a zatim i elementi matrice. Na standardni izlaz ispisati trag matrice.

Primer 1

```
INTERAKCIJA PROGRAMA:
 Unesite broj vrsta i broj kolona matrice:
 2 3
 Unesite elemente matrice, vrstu po vrstu:
 1.2 2.3 3.4
 4.5 5.6 6.7
 Trag unete matrice je 6.80.
```

[Rešenje 1.18]

Zadatak 1.19 Data je celobrojna matrica dimenzije $n \times m$.

(a) Napisati funkciju koja vrši učitavanje matrice sa standardnog ulaza.

(b) Napisati funkciju koja ispisuje elemente ispod glavne dijagonale matrice (uključujući i glavnu dijagonalu).

Napisati program koji testira napisane funkcije. 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.

Primer 1

```
Interakcija programa:
 Unesite broj vrsta i broj kolona matrice:
 2 3
 Unesite elemente matrice, vrstu po vrstu:
 1 -2 3
 -4 5 -6
 Elementi ispod glavne dijagonale matrice:
 1
 -4 5
```

[Rešenje 1.19]

Zadatak 1.20 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.

Primer 1

```
Interakcija programa:
 Unesite broj vrsta i broj kolona matrice:
 2 3
 Unesite elemente matrice, vrstu po vrstu:
 1 2 3
 4 5 6
 Kolona pod rednim brojem 3 ima najveci zbir.
```

Primer 2

```
Interakcija programa:
 Unesite broj vrsta i broj kolona matrice:
 2 4
 Unesite elemente matrice, vrstu po vrstu:
 1 2 3 4
 8 7 6 5
 Kolona pod rednim brojem 1 ima najveci zbir.
```

Zadatak 1.21 Data je realna kvadratna matrica dimenzije 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

```
Poziv: ./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 programa:
 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
```

[Rešenje 1.21]

Zadatak 1.22 Napisati program koji na osnovu dve realne matrice dimenzija $m \times n$ 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 se nalaze dimenzije matrica m i n, 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

```
Poziv: ./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 programa:
 Trazena matrica je:
 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 1.23 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.

Primer 1

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

Zadatak 1.24 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: *Za realokaciju memorije koristiti* `realloc()` *funkciju.*

Primer 1

```
SLICICE.TXT
 3 Brazil
 6 Nemacka
 2 Kamerun
 1 Brazil
 2 Engleska
 4 Engleska
 5 Brazil

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

**** Zadatak 1.25** 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 PROGRAMA:
 U datoteci su zadata temena cetvorougla.
 Obim je 8.
 Povrsina je 4.
```

1.4 Pokazivači na funkcije

Zadatak 1.26 Napisati program koji tabelarno štampa vrednosti proizvoljne realne funkcije sa jednim realnim argumentom, odnosno izračunava i ispisuje vrednosti date funkcije na diskretnoj ekvidistantnoj mreži od n tačaka intervala $[a, b]$. Realni brojevi a i b ($a < b$) kao i ceo broj n ($n \geq 2$) 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

```
Poziv: ./a.out sin

Interakcija programa:
 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

```
Poziv: ./a.out cos

Interakcija programa:
 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 |
 ----------------------
```

[Rešenje 1.26]

Zadatak 1.27 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 programa:
 Unesite ime funkcije, n i a:
 tan 1.570795 10000
 Limes funkcije tan je -10134.5.
```

Zadatak 1.28 Napisati funkciju koja određuje integral funkcije `f(x)` na intervalu $[a, b]$. Adresa funkcije `f` se prenosi kao parametar. Integral se računa

prema formuli:

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

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 programa:
 Unesite ime funkcije, n, a i b:
 cos 6000 -1.5 3.5
 Vrednost integrala je 0.645931.
```

Zadatak 1.29 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 programa:
 Unesite ime funkcije, n, a i b:
 sin 100 -1.0 3.0
 Vrednost integrala je 1.530295.
```

1.5 Rešenja

Rešenje 1.1

```
1 #include <stdio.h>
  #include <stdlib.h>
3
  #define MAX 100
5
  /* Funkcija obrce elemente niza koriscenjem indekse sintakse */
```

```c
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;
  }
}

/* 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, sto za posledicu ima da pokazivac
       "poslednji" sada 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];
```

```
59    /* Broj elemenata niza a */
      int n;
61
      /* Pokazivac na elemente niza */
63    int *p;

65    printf("Unesite dimenziju niza: ");
      scanf("%d", &n);
67
      /* Proverava se da li je doslo do prekoracenja ogranicenja
69       dimenzije */
      if (n <= 0 || n > MAX) {
71      fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
        exit(EXIT_FAILURE);
73    }

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

79    obrni_niz_v1(a, n);

81    printf("Nakon obrtanja elemenata, niz je:\n");

83    for (p = a; p - a < n; p++)
        printf("%d ", *p);
85    printf("\n");

87    obrni_niz_v2(a, n);

89    printf("Nakon ponovnog obrtanja elemenata, niz je:\n");

91    for (p = a; p - a < n; p++)
        printf("%d ", *p);
93    printf("\n");

95    return 0;
  }
```

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++;
24
      return p;
26 }

28 /* Funkcija izracunava minimalni element niza */
double min(double *a, int n)
30 {
      /* Na pocetku, minimalni element je prvi element */
32    double min = a[0];
      int i;
34
      /* Ispituje se da li se medju ostalim elementima niza nalazi
36       minimalni */
      for (i = 1; i < n; i++)
38      if (a[i] < min)
          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 */
48    double max = *a;

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

56    return max;
}
58

60 int main()
{
62    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));

  return 0;
}
```

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) {

    /* Povecava se vrednost elementa na koji pokazuje pokazivac
       prvi */
    (*prvi)++;

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

    /* Smanjuje se vrednost elementa na koji pokazuje pokazivac
       poslednji */
```

```
25      (*poslednji)--;

27      /* Pokazivac poslednji se pomera na prethodni element */
        poslednji--;
29    }

31    /* Drugi nacin */
      while (prvi < poslednji) {
33      (*prvi++)++;
        (*poslednji--)--;
35    }
  }
37
  int main()
39{
    int a[MAX];
41  int n;
    int *p;
43
    printf("Unesite dimenziju niza: ");
45  scanf("%d", &n);

47  /* Proverava se da li je doslo do prekoracenja ogranicenja
       dimenzije */
49  if (n <= 0 || n > MAX) {
      fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
51    exit(EXIT_FAILURE);
    }
53
    printf("Unesite elemente niza:\n");
55  for (p = a; p - a < n; p++)
      scanf("%d", p);
57
    povecaj_smanji(a, n);
59
    printf("Transformisan niz je:\n");
61  for (p = a; p - a < n; p++)
      printf("%d ", *p);
63  printf("\n");

65  return 0;
  }
```

Rešenje 1.4

```
  #include <stdio.h>
2
  int main(int argc, char *argv[])
4 {
    int i;
6   char tip_ispisa;
```

```
  printf("Broj prihvacenih argumenata komandne linije je %d.\n", argc
    );

  printf("Kako zelite da ispisete argumente, ");
  printf("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 su:\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>
```

```c
#define MAX 100

/* Funkcija ispituje da li je niska palindrom */
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;
}

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 komandne linije koji su palindromi je %d.\n
    ", n);
  return 0;
}
```

Rešenje 1.6

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

#define MAX_KARAKTERA 100

/* Implementacija funkcija 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];
  int br = 0, n;
  FILE *in;

  /* Ako korisnik nije uneo trazene argumente, prijavljuje se
     greska */
  if (argc < 3) {
    printf("Greska: ");
```

```
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("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);
  return 0;
}
```

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++);
  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)
{
  if (duzina_niske(sufiks) <= duzina_niske(niska) &&
      poredjenje_niski(niska + duzina_niske(niska) -
                       duzina_niske(sufiks), sufiks) == 0)
    return 1;
  return 0;
}

/* Funkcija ispituje da li je niska zadata drugim argumentom
   funkcije prefiks niske zadate prvi argumentom funkcije */
int prefiks_niske(char *niska, char *prefiks)
{
  int i;
  if (duzina_niske(prefiks) <= duzina_niske(niska)) {
    for (i = 0; i < duzina_niske(prefiks); i++)
      if (*(prefiks + i) != *(niska + i))
        return 0;
    return 1;
  } else
    return 0;
}

int main(int argc, char **argv)
{
  /* Ukoliko korisnik nije uneo trazene argumente, prijavljuje se
     greska */
  if (argc < 4) {
    printf("Greska: ");
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_dat suf/pref -s/-p.\n",
           argv[0]);
    exit(EXIT_FAILURE);
  }

  FILE *in;
  int br = 0;
```

```
  char rec[MAX_KARAKTERA];

  in = fopen(*(argv + 1), "r");
  if (in == NULL) {
    fprintf(stderr, "Greska: ");
    fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
            argv[1]);
    exit(EXIT_FAILURE);
  }

  /* Provera 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);
  return 0;
}
```

Rešenje 1.8

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

#define MAX 100

/* Deklarisemo funkcije koje cemo kasnije da definisemo */
double euklidska_norma(int M[][MAX], int n);
int trag(int M[][MAX], int n);
int gornja_vandijagonalna_norma(int M[][MAX], int n);

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

  /* Unosimo dimenziju kvadratne matrice */
  scanf("%d", &n);

  /* Proveravamo da li je prekoraceno ogranicenje */
  if (n > MAX || n <= 0) {
```

```
    fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
    fprintf(stderr, "matrice.\n");
    exit(EXIT_FAILURE);
  }

  /* Popunjavamo vrstu po vrstu matrice */
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", &A[i][j]);

  /* Ispis elemenata matrice koriscenjem indeksne sintakse.
     Ispis vrsimo vrstu po vrstu */
  for (i = 0; i < n; i++) {
    /* Ispisujemo elemente i-te vrste */
    for (j = 0; j < n; j++)
      printf("%d ", A[i][j]);
    printf("\n");
  }

  /* Ispis elemenata 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"); } */

  int tr = trag(A, n);
  printf("trag = %d\n", tr);

  printf("euklidska norma = %.2f\n", euklidska_norma(A, n));
  printf("vandijagonalna norma = %d\n",
         gornja_vandijagonalna_norma(A, n));

  return 0;
}

/* Definisemo funkcije koju smo ranije deklarisali */

/* 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 */
```

```c
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];

  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;
}
```

Rešenje 1.9

```c
#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
   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++)
      /* Nasli smo elemente na istim pozicijama u matricama koji
         se razlikuju */
      if (a[i][j] != b[i][j])
        return 0;

  /* Prosla je provera jednakosti za sve parove elemenata koji
     su na istim pozicijama sto 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++) {
      /* Mnozimo i-tu vrstu 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], c[MAX][MAX];
```

```
77    /* Matrice zbira i proizvoda */
      int zbir[MAX][MAX], proizvod[MAX][MAX];
79
      /* Dimenzija matrica */
81    int n;
      int i, j;
83
      /* Ucitavamo dimenziju kvadratnih matrica i proveravamo njenu
85       korektnost */
      scanf("%d", &n);
87
      /* Proveravamo da li je prekoraceno ogranicenje */
89    if (n > MAX || n <= 0) {
        fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
91      fprintf(stderr, "matrica.\n");
        exit(EXIT_FAILURE);
93    }

95    /* Ucitavamo matrice */
      ucitaj_matricu(a, n);
97    ucitaj_matricu(b, n);

99    /* Izracunavamo zbir i proizvod matrica */
      saberi(a, b, zbir, n);
101   pomnozi(a, b, proizvod, n);

103   /* Ispisujemo rezultat */
      if (jednake_matrice(a, b, n) == 1)
105     printf("da\n");
      else
107     printf("ne\n");

109   printf("Zbir matrica je:\n");
      ispisi_matricu(zbir, n);
111
      printf("Proizvod matrica je:\n");
113   ispisi_matricu(proizvod, n);

115   return 0;
  }
```

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 sam sa sobom,
```

```
   odnosno ako se u matrici relacije na glavnoj dijagonali nalaze
   jedinice */
int refleksivnost(int m[][MAX], int n)
{
  int i;

  /* Obilazimo glavnu dijagonalu matrice. Za elemente na glavnoj
     dijagonali vazi da je indeks vrste jednak indeksu kolone */
  for (i = 0; i < n; i++) {
    if (m[i][i] != 1)
      return 0;
  }

  return 1;
}

/* 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, j;

  /* Prepisujemo vrednosti elemenata matrice pocetne matrice */
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      zatvorenje[i][j] = m[i][j];

  /* Postavljamo na glavnoj dijagonali 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;

  /* Obilazimo elemente ispod glavne dijagonale matrice i
     uporedjujemo ih 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;
}
```

```
60 /* Funkcija odredjuje simetricno zatvorenje zadate relacije. Ono
      je odredjeno matricom koja sadrzi sve elemente polazne
62    matrice dopunjene tako da matrica postane simetricna u odnosu
      na glavnu dijagonalu */
64 void sim_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
   {
66   int i, j;

68   /* Prepisujemo vrednosti elemenata matrice m */
     for (i = 0; i < n; i++)
70     for (j = 0; j < n; j++)
         zatvorenje[i][j] = m[i][j];
72
     /* Odredjujemo simetricno zatvorenje matrice */
74   for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
76       if (zatvorenje[i][j] == 1)
           zatvorenje[j][i] = 1;
78 }

80
   /* Funkcija proverava da li je relacija tranzitivna. Relacija je
82    tranzitivna ako ispunjava sledece svojstvo: ako je element
      "i" u relaciji sa elementom "j" i element "j" u relaciji sa
84    elementom "k", onda je i element "i" u relaciji sa elementom
      "k" */
86 int tranzitivnost(int m[][MAX], int n)
   {
88   int i, j, k;

90   for (i = 0; i < n; i++)
       for (j = 0; j < n; j++)
92       /* Pokusavamo da pronadjemo element koji narusava *
            tranzitivnost */
94       for (k = 0; k < n; k++)
           if (m[i][k] == 1 && m[k][j] == 1 && m[i][j] == 0)
96           return 0;

98   return 1;
   }
100

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

108   /* Kopiramo pocetnu matricu u matricu rezultata */
      for (i = 0; i < n; i++)
110     for (j = 0; j < n; j++)
          zatvorenje[i][j] = m[i][j];
```

```
112
   /* Primenom Varsalovog algoritma odredjujemo
114      refleksivno-tranzitivno zatvorenje matrice */
   for (k = 0; k < n; k++)
116   for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
118       if ((zatvorenje[i][k] == 1) && (zatvorenje[k][j] == 1)
              && (zatvorenje[i][j] == 0))
120         zatvorenje[i][j] = 1;
}
122
/* Funkcija ispisuje elemente matrice */
124 void pisi_matricu(int m[][MAX], int n)
{
126   int i, j;

128   for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++)
130       printf("%d ", m[i][j]);
      printf("\n");
132   }
}
134
int main(int argc, char *argv[])
136 {
   FILE *ulaz;
138   int m[MAX][MAX];
   int pomocna[MAX][MAX];
140   int n, i, j, k;

142   /* Ako korisnik nije uneo trazene argumente, prijavljujemo
         gresku */
144   if (argc < 2) {
        printf("Greska: ");
146     printf("Nedovoljan broj argumenata komandne linije.\n");
        printf("Program se poziva sa %s ime_dat.\n", argv[0]);
148     exit(EXIT_FAILURE);
      }
150
      /* Otvaramo datoteku za citanje */
152   ulaz = fopen(argv[1], "r");
      if (ulaz == NULL) {
154     fprintf(stderr, "Greska: ");
        fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
156             argv[1]);
        exit(EXIT_FAILURE);
158   }

160   /* Ucitavamo dimenziju matrice */
      fscanf(ulaz, "%d", &n);
162
      /* Proveravamo da li je prekoraceno ogranicenje */
```

```
164   if (n > MAX || n <= 0) {
        fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
166     fprintf(stderr, "matrice.\n");
        exit(EXIT_FAILURE);
168   }

170   /* Ucitavamo element po element matrice */
      for (i = 0; i < n; i++)
172     for (j = 0; j < n; j++)
          fscanf(ulaz, "%d", &m[i][j]);
174
      /* Ispisujemo trazene vrednosti */
176   printf("Refleksivnost: %s\n",
             refleksivnost(m, n) == 1 ? "da" : "ne");
178
      printf("Simetricnost: %s\n",
180          simetricnost(m, n) == 1 ? "da" : "ne");

182   printf("Tranzitivnost: %s\n",
             tranzitivnost(m, n) == 1 ? "da" : "ne");
184
      printf("Refleksivno zatvorenje:\n");
186   ref_zatvorenje(m, n, pomocna);
      pisi_matricu(pomocna, n);
188
      printf("Simetricno zatvorenje:\n");
190   sim_zatvorenje(m, n, pomocna);
      pisi_matricu(pomocna, n);
192
      printf("Refleksivno-tranzitivno zatvorenje:\n");
194   tran_zatvorenje(m, n, pomocna);
      pisi_matricu(pomocna, n);
196
      /* Zatvaramo datoteku */
198   fclose(ulaz);

200   return 0;
}
```

Rešenje 1.11

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

4 #define MAX 32

6 int max_sporedna_dijagonala(int m[][MAX], int n)
{
8   int i, j;
    /* Trazimo najveci element na sporednoj dijagonali. Za
10     elemente sporedne dijagonale vazi da je zbir indeksa vrste
```

```
     i indeksa kolone jednak n-1. Za pocetnu vrednost maksimuma
     uzimamo element u gornjem desnom uglu */
  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 */
int indeks_min(int m[][MAX], int n)
{
  int i, j;
  /* Za pocetnu vrednost minimuma uzimamo element u gornjem
     levom uglu */
  int min = m[0][0], indeks_kolone = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      /* Ako je tekuci element manji od minimalnog */
      if (m[i][j] < min) {
        /* cuvamo njegovu vrednost */
        min = m[i][j];
        /* i cuvamo indeks kolone u kojoj se nalazi */
        indeks_kolone = j;
      }
  return indeks_kolone;
}

/* Funkcija izracunava indeks vrste najveceg elementa */
int indeks_max(int m[][MAX], int n)
{
  int i, j;
  /* Za maksimalni element uzimamo gornji levi ugao */
  int max = m[0][0], indeks_vrste = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      /* Ako je tekuci element manji od minimalnog */
      if (m[i][j] > max) {
        /* cuvamo njegovu vrednost */
        max = m[i][j];
        /* i cuvamo indeks vrste u kojoj se nalazi */
        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;

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

  /* Ucitavamo vrednost dimenzije i proveravamo njenu korektnost
   */
  n = atoi(argv[1]);

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

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

  int max_sd = max_sporedna_dijagonala(m, n);
  int i_min = indeks_min(m, n);
  int i_max = indeks_max(m, n);
  int bn = broj_negativnih(m, n);

  /* Ispisujemo rezultat */
  printf("%d %d %d %d\n", max_sd, i_min, i_max, bn);

  /* Prekidamo izvrsavanje programa */
  return 0;
}
```

Rešenje 1.12

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

#define MAX 32

/* Funkcija ucitava elemente kvadratne matrice 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 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 */
int ortonormirana(int m[][MAX], int n)
{
  int i, j, k;
  int proizvod;

  /* Proveravamo uslov normiranosti, odnosno da li je proizvod
     svake vrste matrice sa samom sobom jednak jedinici */
  for (i = 0; i < n; i++) {

    /* Izracunavamo skalarni proizvod vrste sa samom sobom */
    proizvod = 0;

    for (j = 0; j < n; j++)
      proizvod += m[i][j] * m[i][j];

    /* Ako proizvod bar jedne vrste nije jednak jedinici, odmah
       zakljucujemo da matrica nije normirana */
    if (proizvod != 1)
      return 0;
  }
```

```
51
   /* Proveravamo uslov ortogonalnosti, odnosno da li je proizvod
53    dve bilo koje razlicite vrste matrice jednak nuli */
   for (i = 0; i < n - 1; i++) {
55   for (j = i + 1; j < n; j++) {

57     /* Izracunavamo skalarni proizvod */
       proizvod = 0;
59
       for (k = 0; k < n; k++)
61       proizvod += m[i][k] * m[j][k];

63     /* Ako proizvod dve bilo koje razlicite vrste nije jednak
          nuli, odmah zakljucujemo da matrica nije ortogonalna */
65     if (proizvod != 0)
         return 0;
67   }
   }
69
   /* Ako su oba uslova ispunjena, vracamo jedinicu kao rezultat */
71 return 1;
}
73
int main()
75{
   int A[MAX][MAX];
77 int n;

79 /* Ucitavamo vrednost dimenzije i proveravamo njenu korektnost
    */
81 scanf("%d", &n);

83 if (n > MAX || n <= 0) {
     fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
85   fprintf(stderr, "matrice.\n");
     exit(EXIT_FAILURE);
87 }

89 /* Ucitavamo matricu */
   ucitaj_matricu(A, n);
91
   /* Ispisujemo rezultat rada funkcije */
93 if (ortonormirana(A, n))
     printf("da\n");
95 else
     printf("ne\n");
97
   return 0;
99}
```

Rešenje 1.13

```c
#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 krajIspisa(int top, int bottom, int left, int right)
{
  return !(top <= bottom && left <= right);
}

/* Funkcija spiralno ispisuje elemente matrice */
void ispisi_matricu_spiralno(int a[][MAX_K], int n, int m)
{
  int i, j, top, bottom, left, right;

  top = left = 0;
  bottom = n - 1;
  right = m - 1;

  while (!krajIspisa(top, bottom, left, right)) {
    /* Ispisuje se prvi red */
    for (j = left; j <= right; j++)
      printf("%d ", a[top][j]);

    /* Spustamo prvi red */
    top++;

    if (krajIspisa(top, bottom, left, right))
      break;

    for (i = top; i <= bottom; i++)
      printf("%d ", a[i][right]);

    /* Pomeramo desnu kolonu za naredni krug ispisa blize levom
       kraju */
    right--;

    if (krajIspisa(top, bottom, left, right))
      break;

    /* Ispisujemo donju vrstu */
    for (j = right; j >= left; j--)
      printf("%d ", a[bottom][j]);

    /* Podizemo donju vrstu za naredni krug ispisa */
    bottom--;

    if (krajIspisa(top, bottom, left, right))
```

```
      break;

    /* Ispisujemo prvu kolonu */
    for (i = bottom; i >= top; i--)
      printf("%d ", a[i][left]);

    /* Pripremamo levu kolonu za naredni krug ispisa */
    left++;
  }
  putchar('\n');
}

void ucitaj_matricu(int a[][MAX_K], int n, int m)
{
  int i, j;

  for (i = 0; i < n; i++)
    for (j = 0; j < m; j++)
      scanf("%d", &a[i][j]);
}

int main()
{
  int a[MAX_V][MAX_K];
  int m, n;

  /* Ucitaj broj vrsta i broj kolona matrice */
  scanf("%d", &n);
  scanf("%d", &m);

  if (n > MAX_V || n <= 0 || m > MAX_K || m <= 0) {
    fprintf(stderr, "Greska: neodgovarajuce dimenzije ");
    fprintf(stderr, "matrice.\n");
    exit(EXIT_FAILURE);
  }

  ucitaj_matricu(a, n, m);
  ispisi_matricu_spiralno(a, n, m);

  return 0;
}
```

Rešenje 1.15

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

/* NAPOMENA: Primer demonstrira dinamicku alokaciju niza od n
   elemenata. Dovoljno je alocirati n * sizeof(T) bajtova, gde
   je T tip elemenata niza. Povratnu adresu malloc()-a treba
   pretvoriti iz void * u T *, kako bismo dobili pokazivac koji
```

```
   pokazuje na prvi element niza tipa T. Na dalje se elementima
   moze pristupati na isti nacin kao da nam je dato ime niza
   (koje se tako i ponasa - kao pokazivac na element tipa T koji
   je prvi u nizu) */
int main()
{
  int *p = NULL;
  int i, n;

  /* Unosimo dimenziju niza. Ova vrednost nije ogranicena bilo
     kakvom konstantom, kao sto je to ranije bio slucaj kod
     staticke alokacije gde je dimenzija niza bila unapred
     ogranicena definisanim prostorom. */
  scanf("%d", &n);

  /* Alociramo prostor za n celih brojeva */
  if ((p = (int *) malloc(sizeof(int) * n)) == NULL) {
    fprintf(stderr, "malloc(): ");
    fprintf(stderr, "greska pri alokaciji memorije.\n");
    exit(EXIT_FAILURE);
  }

  /* Od ovog trenutka pokazivac "p" mozemo da koristimo kao da
     je ime niza, odnosno i-tom elementu se moze pristupiti sa
     p[i] */

  /* Unosimo elemente niza */
  for (i = 0; i < n; i++)
    scanf("%d", &p[i]);

  /* Ispisujemo elemente niza unazad */
  for (i = n - 1; i >= 0; i--)
    printf("%d ", p[i]);
  printf("\n");

  /* Oslobadjamo prostor */
  free(p);

  return 0;
}
```

Rešenje 1.16

```
#include <stdio.h>
#include <stdlib.h>
#define KORAK 10

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

```
10  /* Velicina alocirane memorije */
    int alocirano;
12
    /* Broj elemenata niza */
14  int n;

16  /* Broj koji se ucitava sa ulaza */
    int x;
18  int i;
    int *b = NULL;
20
    /* Inicijalizacija */
22  alocirano = n = 0;

24  /* Unosimo brojeve sa ulaza */
    scanf("%d", &x);
26
    /* Sve dok je procitani broj razlicit od nule... */
28  while (x != 0) {

30    /* Ako broj ucitanih elemenata niza odgovara broju
         alociranih mesta, za smestanje novog elemenata treba
32       obezbediti dodatni prostor. Da se ne bi za svaki sledeci
         element pojedinacno alocirala memorija, prilikom
34       alokacije se vrsi rezervacija za jos KORAK dodatnih mesta
         za buduce elemente */
36    if (n == alocirano) {
        /* Povecava se broj alociranih mesta */
38      alocirano = alocirano + KORAK;

40      /* Vrsi se realokacija memorije sa novom velicinom */
        /*******************************************************/
42      /* Resenje sa funkcijom malloc() */
        /*******************************************************/
44      /* Vrsi se alokacija memorije sa novom velicinom, a adresa
           pocetka novog memorijskog bloka se cuva u promenljivoj
46         b */
        b = (int *) malloc(alocirano * sizeof(int));
48
        /* Ako prilikom alokacije dodje do neke greske */
50      if (b == NULL) {
          /* poruku ispisujemo na izlaz za greske */
52        fprintf(stderr, "malloc(): ");
          fprintf(stderr, "greska pri alokaciji memorije.\n");
54
          /* Pre kraja programa moramo svu dinamicki alociranu
56           memoriju da oslobodimo. U ovom slucaju samo memoriju
             na adresi a */
58        free(a);

60        /* Zavrsavamo program */
```

```
        exit(EXIT_FAILURE);
      }

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

      /* Posle prepisivanja oslobadjamo blok memorije sa
         pocetnom adresom u a */
      free(a);

      /* Promenljivoj a dodeljujemo adresu pocetka novog, veceg
         bloka koji je prilikom alokacije zapamcen u
         promenljivoj b */
      a = b;

      /*********************************************************/
      /* Resenje sa funkcijom realloc() */
      /*********************************************************/
      /* Zbog funkcije realloc je neophodno da i u prvoj
         iteraciji "a" bude inicijalizovano na NULL */
      /*
         a = (int*) realloc(a,alocirano*sizeof(int));

         if(a == NULL) { fprintf(stderr, "realloc(): ");
         fprintf(stderr, "greska pri alokaciji memorije.\n");
         exit(EXIT_FAILURE); } */
    }

    /* Smestamo element u niz */
    a[n++] = x;

    /* i ucitavamo sledeci element */
    scanf("%d", &x);
  }

  /* Ispisujemo brojeve u obrnutom poretku */
  for (n--; n >= 0; n--)
    printf("%d ", a[n]);
  printf("\n");

  /* Oslobadjamo dinamicki alociranu memoriju */
  free(a);

  /* Program se zavrsava */
  exit(EXIT_SUCCESS);
}
```

Rešenje 1.17

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

#define MAX 1000

/* NAPOMENA: Primer demonstrira "vracanje nizova iz funkcije".
   Ovako nesto se moze improvizovati tako sto se u funkciji
   dinamicki kreira niz potrebne velicine, popuni se potrebnim
   informacijama, a zatim se vrati njegova adresa. Imajuci u
   vidu cinjenicu da dinamicki kreiran objekat ne nestaje kada
   se izadje iz funkcije koja ga je kreirala, vraceni pokazivac
   se kasnije u pozivajucoj funkciji moze koristiti za pristup
   "vracenom" nizu. Medjutim, pozivajuca funkcija ima
   odgovornost i da se brine o dealokaciji istog prostora */

/* Funkcija dinamicki kreira niz karaktera u koji smesta
   rezultat nadovezivanja niski. Adresa niza se vraca kao
   povratna vrednost. */
char *nadovezi(char *s, char *t)
{
  /* Dinamicki kreiramo prostor dovoljne velicine */
  char *p = (char *) malloc((strlen(s) + strlen(t) + 1)
                            * sizeof(char));

  /* Proveravamo uspeh alokacije */
  if (p == NULL) {
    fprintf(stderr, "malloc(): ");
    fprintf(stderr, "greska pri alokaciji memorije.\n");
    exit(EXIT_FAILURE);
  }

  /* Kopiramo i nadovezujemo stringove */

  /* Resenje bez koriscenja biblioteckih funkcija */
  /*
     int i,j; for(i=j=0; s[j]!='\0'; i++, j++) p[i]=s[j];

     for(j=0; t[j]!='\0'; i++, j++) p[i]=t[j];

     p[i]='\0'; */

  /* Resenje sa koriscenjem biblioteckih funkcija iz zaglavlja
     string.h */
  strcpy(p, s);
  strcat(p, t);

  /* Vracamo pokazivac p */
  return p;
}
```

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

  /* Ucitavamo dve niske koje cemo da nadovezemo */
  scanf("%s", s1);
  scanf("%s", s2);

  /* Pozivamo funkciju da nadoveze stringove */
  s = nadovezi(s1, s2);

  /* Prikazujemo rezultat */
  printf("%s\n", s);

  /* Oslobadjamo memoriju alociranu u funkciji nadovezi() */
  free(s);

  return 0;
}
```

Rešenje 1.18

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

int main()
{
  int i, j;

  /* Pokazivac na dinamicki alociran niz pokazivaca na vrste
     matrice */
  double **A = NULL;

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

  /* Trag matice */
  double trag = 0;

  /* Unosimo dimenzije matrice */
  scanf("%d%d", &n, &m);

  /* Dinamicki alociramo prostor za n pokazivaca na double */
  A = malloc(sizeof(double *) * n);

  /* Proveramo da li je doslo do greske pri alokaciji */
  if (A == NULL) {
    fprintf(stderr, "malloc(): ");
    fprintf(stderr, "greska pri alokaciji memorije.\n");
```

```
29      exit(EXIT_FAILURE);
    }
31
    /* Dinamicki alociramo prostor za elemente u vrstama */
33  for (i = 0; i < n; i++) {
      A[i] = malloc(sizeof(double) * m);
35
      if (A[i] == NULL) {
37      /* Alokacija je neuspesna. Pre zavrsetka programa moramo
           da oslobodimo svih i-1 prethodno alociranih vrsta, i
39         alociran niz pokazivaca */
        for (j = 0; j < i; j++)
41        free(A[j]);
        free(A);
43
        exit(EXIT_FAILURE);
45    }
    }
47
    /* Unosimo sa standardnog ulaza brojeve u matricu. Popunjavamo
49     vrstu po vrstu */
    for (i = 0; i < n; i++)
51    for (j = 0; j < m; j++)
        scanf("%lf", &A[i][j]);
53
    /* Racunamo trag matrice, odnosno sumu elemenata na glavnoj
55     dijagonali */
    trag = 0.0;
57
    for (i = 0; i < n; i++)
59    trag += A[i][i];

61  printf("%.2f\n", trag);

63  /* Oslobadjamo prostor rezervisan za svaku vrstu */
    for (j = 0; j < n; j++)
65    free(A[j]);

67  /* Oslobadjamo memoriju za niz pokazivaca na vrste */
    free(A);
69
    return 0;
71  }
```

Rešenje 1.19

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

5  void ucitaj_matricu(int **M, int n, int m)
```

```
{
  int i, j;

  /* Popunjavamo matricu vrstu po vrstu */
  for (i = 0; i < n; i++)
    /* Popunjavamo i-tu vrstu matrice */
    for (j = 0; j < m; j++)
      scanf("%d", &M[i][j]);
}

void ispisi_elemente_ispod_dijagonale(int **M, int n, int m)
{
  int i, j;

  for (i = 0; i < n; i++) {
    /* Ispisujemo elemente ispod glavne dijagonale matrice */
    for (j = 0; j <= i; j++)
      printf("%d ", M[i][j]);
    printf("\n");
  }
}

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

  /* Unosimo dimenzije matrice */
  scanf("%d %d", &n, &m);

  /* Alociramo prostor za niz pokazivaca na vrste matrice */
  matrica = (int **) malloc(n * sizeof(int *));
  if (matrica == NULL) {
    fprintf(stderr, "malloc(): Neuspela alokacija\n");
    exit(EXIT_FAILURE);
  }

  /* Alociramo prostor za svaku vrstu matrice */
  for (i = 0; i < n; i++) {
    matrica[i] = (int *) malloc(m * sizeof(int));

    if (matrica[i] == NULL) {
      fprintf(stderr, "malloc(): Neuspela alokacija\n");
      for (j = 0; j < i; j++)
        free(matrica[j]);
      free(matrica);
      exit(EXIT_FAILURE);
    }
  }

  ucitaj_matricu(matrica, n, m);

```

```
  ispisi_elemente_ispod_dijagonale(matrica, n, m);

  /* Oslobadjamo dinamicki alociranu memoriju za matricu. Prvo
     oslobadjamo prostor rezervisan za svaku vrstu */
  for (j = 0; j < n; j++)
    free(matrica[j]);

  /* Zatim oslobadjamo memoriju za niz pokazivaca na vrste
     matrice */
  free(matrica);

  return 0;
}
```

Rešenje 1.21

```
#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++)
      /* Ako je indeks vrste manji od indeksa kolone */
      if (i < j)
        /* element se nalazi iznad glavne dijagonale pa ga
           polovimo */
        a[i][j] /= 2;
      else
        /* Ako je indeks vrste veci od indeksa kolone */
      if (i > j)
        /* element se nalazi ispod glavne dijagonale pa ga
           dupliramo */
        a[i][j] *= 2;
}

/* Funkcija izracunava zbir apsolutnih vrednosti elemenata ispod
   sporedne dijagonale */
float zbir_ispod_sporedne_dijagonale(float **m, int n)
{
  int i, j;
  float zbir = 0;

  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      /* Ukoliko je zbir indeksa vrste i indeksa kolone elementa
         veci od n-1, to znaci da se element nalazi ispod
```

```
36          sporedne dijagonale */
      if (i + j > n - 1)
38        zbir += fabs(m[i][j]);

40  return zbir;
}
42
/* Funkcija ucitava elemente kvadratne matrice dimenzije n iz
44    zadate datoteke */
void ucitaj_matricu(FILE * ulaz, float **m, int n)
46 {
  int i, j;
48
  for (i = 0; i < n; i++)
50    for (j = 0; j < n; j++)
      fscanf(ulaz, "%f", &m[i][j]);
52 }
54 /* Funkcija ispisuje elemente kvadratne matrice dimenzije n na
   standardni izlaz */
56 void ispisi_matricu(float **m, int n)
{
58  int i, j;

60  for (i = 0; i < n; i++) {
    for (j = 0; j < n; j++)
62    printf("%.2f ", m[i][j]);
    printf("\n");
64  }
}
66
/* Funkcija alocira memoriju za kvadratnu matricu dimenzije n */
68 float **alociraj_memoriju(int n)
{
70  int i, j;
  float **m;
72
  m = (float **) malloc(n * sizeof(float *));
74  if (m == NULL) {
    fprintf(stderr, "malloc(): Neuspela alokacija\n");
76    exit(EXIT_FAILURE);
  }
78
  /* Za svaku vrstu matrice */
80  for (i = 0; i < n; i++) {
    /* Alociramo memoriju */
82    m[i] = (float *) malloc(n * sizeof(float));

84    /* Proveravamo da li je doslo do greske pri alokaciji */
    if (m[i] == NULL) {
86      /* Ako jeste, ispisujemo poruku */
      printf("malloc(): neuspela alokacija memorije!\n");
```

```
88
      /* Oslobadjamo memoriju zauzetu do ovog koraka */
90    for (j = 0; j < i; j++)
        free(m[i]);
92    free(m);
      exit(EXIT_FAILURE);
94  }
  }
96 return m;
}
98
/* Funckija oslobadja memoriju zauzetu kvadratnom matricom
100   dimenzije n */
void oslobodi_memoriju(float **m, int n)
102 {
  int i;
104
  for (i = 0; i < n; i++)
106   free(m[i]);
  free(m);
108 }

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

116 /* Ako korisnik nije uneo trazene argumente, prijavljujemo
      gresku */
118 if (argc < 2) {
    printf("Greska: ");
120   printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_dat.\n", argv[0]);
122   exit(EXIT_FAILURE);
  }
124
  /* Otvaramo datoteku za citanje */
126 ulaz = fopen(argv[1], "r");
  if (ulaz == NULL) {
128   fprintf(stderr, "Greska: ");
    fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
130           argv[1]);
    exit(EXIT_FAILURE);
132 }

134 /* citamo dimenziju matrice */
  fscanf(ulaz, "%d", &n);
136
  /* Alociramo memoriju */
138 a = alociraj_memoriju(n);
```

```
140   /* Ucitavamo elemente matrice */
      ucitaj_matricu(ulaz, a, n);
142
      float zbir = zbir_ispod_sporedne_dijagonale(a, n);
144
      /* Pozivamo funkciju za modifikovanje elemenata */
146   izmeni(a, n);

148   /* Ispisujemo rezultat */
      printf("Zbir apsolutnih vrednosti ispod sporedne dijagonale ");
150   printf("je %.2f.\n", zbir);

152   printf("Transformisana matrica je:\n");
      ispisi_matricu(a, n);
154
      /* Oslobadjamo memoriju */
156   oslobodi_memoriju(a, n);

158   /* Zatvaramo datoteku */
      fclose(ulaz);
160
      /* i prekidamo sa izvrsavanjem programa */
162   return 0;
}
```

Rešenje 1.22

Rešenje 1.26

```
1
#include <stdio.h>
3 #include <stdlib.h>
#include <math.h>
5 #include <string.h>

7 /* NAPOMENA: Zaglavlje math.h sadrzi deklaracije raznih
     matematickih funkcija. đIzmeu ostalog, to su ćsledee
9    funkcije: double sin(double x); double cos(double x); double
     tan(double x); double asin(double x); double acos(double x);
11   double atan(double x); double atan2(double y, double x);
     double sinh(double x); double cosh(double x); double
13   tanh(double x); double exp(double x); double log(double x);
     double log10(double x); double pow(double x, double y);
15   double sqrt(double x); double ceil(double x); double
     floor(double x); double fabs(double x); */
17
/* Funkcija tabela() prihvata granice intervala a i b, broj
19   ekvidistantnih čtaaka n, kao i čpokaziva f koji pokazuje na
     funkciju koja prihvata double argument, i ćvraa double
21   vrednost. Za tako datu funkciju ispisuje njene vrednosti u
```

```
   intervalu [a,b] u n ekvidistantnih čtaaka 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");
}

/* Umesto da koristimo stepenu funkciju iz zaglavlja math.h ->
   pow(a,2) ćpozivaemo čkorisniku sqr(a) */
double sqr(double a)
{
  return a * a;
}

int main(int argc, char *argv[])
{
  double a, b;
  int n;
  /* Imena funkicja koja ćemo navoditi su ćkraa ili čtano
     duga 5 karaktera */
  char ime_fje[6];
  /* Pokazivac na funkciju koja ima jedan argument tipa double i
     povratnu vrednost istog tipa */
  double (*fp) (double);

  /* Ako korisnik nije uneo žtraene argumente, prijavljujemo
     šgreku */
  if (argc < 2) {
    printf("Greska: ");
    printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_funkcije iz math.h.\n",
           argv[0]);
    exit(EXIT_FAILURE);
  }

  /* Niska ime_fje žsadri ime žtraene funkcije koja je
     navedena u komandnoj liniji */
  strcpy(ime_fje, argv[1]);

  /* Inicijalizujemo čpokaziva na funkciju koja treba da se
     tabelira */
  if (strcmp(ime_fje, "sin") == 0)
    fp = &sin;
  else if (strcmp(ime_fje, "cos") == 0)
    fp = &cos;
```

```
  else if (strcmp(ime_fje, "tan") == 0)
75  fp = &tan;
  else if (strcmp(ime_fje, "atan") == 0)
77  fp = &atan;
  else if (strcmp(ime_fje, "acos") == 0)
79  fp = &acos;
  else if (strcmp(ime_fje, "asin") == 0)
81  fp = &asin;
  else if (strcmp(ime_fje, "exp") == 0)
83  fp = &exp;
  else if (strcmp(ime_fje, "log") == 0)
85  fp = &log;
  else if (strcmp(ime_fje, "log10") == 0)
87  fp = &log10;
  else if (strcmp(ime_fje, "sqrt") == 0)
89  fp = &sqrt;
  else if (strcmp(ime_fje, "floor") == 0)
91  fp = &floor;
  else if (strcmp(ime_fje, "ceil") == 0)
93  fp = &ceil;
  else if (strcmp(ime_fje, "sqr") == 0)
95  fp = &sqr;
  else {
97  printf("Program jos uvek ne podrzava trazenu funkciju!\n");
    exit(EXIT_SUCCESS);
99  }

101 printf("Unesite krajeve intervala:\n");
  scanf("%lf %lf", &a, &b);
103
  printf("Koliko tacaka ima na ekvidistantnoj mrezi ");
105 printf("(ukljucujuci krajeve intervala)?\n");
  scanf("%d", &n);
107
  /* Mreza mora da čukljuuje bar krajeve intervala, tako da se
109    mora uneti broj veci od 2 */
  if (n < 2) {
111   fprintf(stderr, "Broj čtaaka žmree mora biti bar 2!\n");
    exit(EXIT_FAILURE);
113 }

115 /* Ispisujemo ime funkcije */
  printf("     x %10s(x)\n", ime_fje);
117
  /* đProsleujemo funkciji tabela() funkciju zadatu kao
119    argument komandne linije */
  tabela(a, b, n, fp);
121
  exit(EXIT_SUCCESS);
123 }
```