

Univerzitet u Beogradu  
Matematički fakultet

Milena, Jelena, Ana, Mirko, Anđelka, Nina

## Zbirka programa

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, ispravan rad sa pokazivačima i dinamički alociranom memorijom, osnovne algoritme pretraživanja i sortiranja, kao i rad 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.

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.

Zahvaljujemo se recenzentima na ..., kao i studentima koji su svojim aktivnim učešćem u nastavi pomogli i doprineli u obličavanju ovog materijala.

*Autori*

---

# Sadržaj

<b>1</b>	<b>Uvodni zadaci</b>	<b>3</b>
1.1	Podela koda po datotekama . . . . .	3
1.2	Algoritmi za rad sa bitovima . . . . .	3
1.3	Rekurzija . . . . .	9
1.4	Rešenja . . . . .	12
<b>2</b>	<b>Pokazivači</b>	<b>15</b>
2.1	Pokazivačka aritmetika . . . . .	15
2.2	Višedimenzioni nizovi . . . . .	18
2.3	Dinamička alokacija memorije . . . . .	22
2.4	Pokazivači na funkcije . . . . .	25
2.5	Rešenja . . . . .	27
<b>3</b>	<b>Algoritmi pretrage i sortiranja</b>	<b>63</b>
3.1	Pretraživanje . . . . .	63
3.2	Sortiranje . . . . .	67
3.3	Bibliotečke funkcije pretrage i sortiranja . . . . .	74
3.4	Rešenja . . . . .	77
<b>4</b>	<b>Dinamičke strukture podataka</b>	<b>89</b>
4.1	Liste . . . . .	89
4.2	Stabla . . . . .	94
4.3	Rešenja . . . . .	102
<b>5</b>	<b>Ispitni rokovi</b>	<b>107</b>
5.1	Programiranje 2, praktični deo ispita, jun 2015. . . . .	107
5.2	Programiranje 2, praktični deo ispita, jul 2015. . . . .	108
5.3	Rešenja . . . . .	110
	<b>Literatura</b>	<b>116</b>



# Glava 1

## Uvodni zadaci

### 1.1 Podela koda po datotekama

### 1.2 Algoritmi za rad sa bitovima

#### Zadatak 1.1

- (a) Napisati funkciju `print_bits` koja štampa bitove u binarnom zapisu celog broja  $x$ .
- (b) Napisati program koja testira funkciju `print_bits` za brojeve koji se sa standardnog ulaza zadaju u heksadekaskom formatu.

#### *Test 1*

```
|| Ulaz:  0x7F
|| Izlaz: 0000 0000 0000 0000 0000 0000 0111 1111
```

#### *Test 2*

```
|| Ulaz:  0x80
|| Izlaz: 0000 0000 0000 0000 0000 0000 1000 0000
```

#### *Test 3*

```
|| Ulaz:  0x00FF00FF
|| Izlaz: 0000 0000 1111 1111 0000 0000 1111 1111
```

#### *Test 4*

```
|| Ulaz:  0xFFFFFFFF
|| Izlaz: 1111 1111 1111 1111 1111 1111 1111 1111
```

#### *Test*

```
|| Ulaz:  0xABCDE123
|| Izlaz: 1010 1011 1100 1101 1110 0001 0010 0011
```

## 1 Uvodni zadaci

---

**Zadatak 1.2** Napisati funkciju koja broji bitove postavljene na 1 u zapisu broja  $x$ . Napisati program koji testira tu funkciju za brojeve koji se sa standardnog ulaza zadaju u heksadekaskom formatu.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<code>Ulaz: 0x7F</code> <code>Izlaz: 7</code>	<code>Ulaz: 0x80</code> <code>Izlaz: 1</code>	<code>Ulaz: 0x00FF00FF</code> <code>Izlaz: 16</code>
<i>Test 4</i>	<i>Test 4</i>	
<code>Ulaz: 0xFFFFFFFF</code> <code>Izlaz: 32</code>	<code>Ulaz: 0xABCDE123</code> <code>Izlaz: 17</code>	

### Zadatak 1.3

- (a) Napisati funkciju **najveci** koja određuje najveći broj koji se može zapisati istim binarnim ciframa kao dati broj i funkciju **najmanji** koja određuje najmanji broj koji se može zapisati istim binarnim ciframa kao dati broj.
- (b) Napisati program koji testira prethodno napisane funkcije tako što prikazuje binarnu reprezentaciju brojeva koji se dobijaju nakon poziva funkcije **najveci**, odnosno **najmanji** za brojeve koji se sa standardnog ulaza zadaju u heksadekaskom formatu.

*Test 1*

```
Ulaz: 0x7F
Izlaz:
Najveci:
1111 1110 0000 0000 0000 0000 0000 0000
Najmanji:
0000 0000 0000 0000 0000 0000 0111 1110
```

*Test 2*

```
Ulaz: 0x80
Izlaz:
Najveci:
1000 0000 0000 0000 0000 0000 0000 0000
Najmanji:
0000 0000 0000 0000 0000 0000 0000 0001
```

*Test 3*

```
Ulaz: 0x00FF00FF
Izlaz:
Najveci:
1111 1111 1111 1111 0000 0000 0000 0000
Najmanji:
0000 0000 0000 0000 1111 1111 1111 1111
```



Test 4

```

Ulaz:  0xFFFFFFFF
Izlaz:
Najveci:
1111 1111 1111 1111 1111 1111 1111 1111
Najmanji:
1111 1111 1111 1111 1111 1111 1111 1111

```

Test 4

```

Ulaz:  0xABCDE123
Izlaz:
Najveci:
1111 1111 1111 1111 1000 0000 0000 0000
Najmanji:
0000 0000 0000 0001 1111 1111 1111 1111

```

**Zadatak 1.4** Napisati program za rad sa bitovima.

- Napisati funkciju koja određuje broj koji se dobija kada se  $n$  bitova datog broja, počevši od pozicije  $p$  postave na 0.
- Napisati funkciju koja određuje broj koji se dobija kada se  $n$  bitova datog broja, počevši od pozicije  $p$  postave na 1.
- Napisati funkciju koja određuje broj koji se dobija kada se  $n$  bitova datog broja, počevši od pozicije  $p$  i vraća ih kao bitove najmanje težine rezultata.
- Napisati funkciju koja vraća broj koji se dobija upisivanjem poslednjih  $n$  bitova broja  $y$  u broj  $x$ , počevši od pozicije  $p$ .
- Napisati funkciju koja vraća broj koji se dobija invertovanjem  $n$  bitova broja  $x$  počevši od pozicije  $p$ .
- Napisati program koji testira prethodno napisane funkcije.

Program treba da testira prethodno napisane funkcije nad neoznačenim celim brojem koji se unosi sa standardnog ulaza. *Napomena: Pozicije se broje počev od pozicije najnižeg bita, pri čemu se broji od nule.*

**Zadatak 1.5**

- Napisati funkciju koja određuje broj koji se dobija rotiranjem u levo datog celog broja. *Napomena: Rotiranje podrazumeva pomeranje svih bitova za jednu poziciju ulevo, s tim što se bit sa pozicije najviše težine pomera na mesto najmanje težine.*
- Napisati funkciju koja određuje broj koji se dobija rotiranjem u desno datog celog neoznačenog broja.

## 1 Uvodni zadaci

---

- (c) Napisati funkciju koja određuje broj koji se dobija rotiranjem u desno datog celog broja.
- (d) Napisati program koji testira prethodno napisane funkcije za brojeve koji se sa standardnog ulaza zadaju u heksadekaskom formatu.

**Zadatak 1.6** Napisati funkciju `mirror` koja određuje ceo broj čiji binarni zapis je slika u ogledalu binarnog zapisa argumenta funkcije. Napisati i program koji testira datu funkciju za brojeve koji se sa standardnog ulaza zadaju u heksadekaskom formatu, tako što najpre ispisuje binarnu reprezentaciju unetog broja, a potom i binarnu reprezentaciju broja dobijenog nakon poziva funkcije `mirror` za uneti broj.

**Zadatak 1.7** Napisati funkciju `int Broj01(unsigned int n)` koja za dati broj `n` vraća 1 ako u njegovom binarnom zapisu ima više jedinica nego nula, a inače vraća 0. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

	<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
	<code>Ulaz:</code> 10	<code>Ulaz:</code> 1024	<code>Ulaz:</code> 2147377146
	<code>Izlaz:</code> 0	<code>Izlaz:</code> 0	<code>Izlaz:</code> 1

  

	<i>Test 4</i>
	<code>Ulaz:</code> 1111111115
	<code>Izlaz:</code> 0

**Zadatak 1.8** Napisati funkciju koja broji koliko se puta kombinacija 11 (dve uzastopne jedinice) pojavljuje u binarnom zapisu celog neoznačenog broja  $x$ . Tri uzastopne jedinice se broje dva puta. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

	<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
	<code>Ulaz:</code> 11	<code>Ulaz:</code> 1024	<code>Ulaz:</code> 2147377146
	<code>Izlaz:</code> 1	<code>Izlaz:</code> 0	<code>Izlaz:</code> 22

  

	<i>Test 4</i>
	<code>Ulaz:</code> 1111111115
	<code>Izlaz:</code> 6

**Zadatak 1.9** Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na pozicijama  $i$ ,  $j$ . Pozicije  $i$ ,  $j$  se učitavaju kao parametri komandne linije. Smatrati da je krajnji desni bit binarne reprezentacije 0-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore  $+$ ,  $-$ ,  $/$ ,  $*$ ,  $\%$ .

Test 1	Test 2	Test 3
<code>Poziv: ./a.out 1 2</code>	<code>Poziv: ./a.out 1 2</code>	<code>Poziv: ./a.out 12 12</code>
<code>Ulaz: 11</code>	<code>Ulaz: 1024</code>	<code>Ulaz: 12345</code>
<code>Izlaz: 13</code>	<code>Izlaz: 1024</code>	<code>Izlaz: 12345</code>

**Zadatak 1.10** Napisati funkciju koja na osnovu neoznačenog broja  $x$  formira nisku  $s$  koja sadrži heksadekadni zapis broja  $x$ , koristeći algoritam za brzo prevođenje binarnog u heksadekadni zapis (svake 4 binarne cifre se zamenjuju jednom odgovarajućom heksadekadnom cifrom). Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1	Test 2	Test 3
<code>Ulaz: 11</code>	<code>Ulaz: 1024</code>	<code>Ulaz: 12345</code>
<code>Izlaz: 0000000B</code>	<code>Izlaz: 00000400</code>	<code>Izlaz: 00003039</code>

**Zadatak 1.11** Napisati funkciju koja za dva data neoznačena broja  $x$  i  $y$  invertuje u podatku  $x$  one bitove koji se poklapaju sa odgovarajućim bitovima u broju  $y$ . Ostali bitovi ostaju nepromenjeni. Napisati program koji tu funkciju testira za brojeve koji se zadaju sa standardnog ulaza.

Test 1	Test 2	Test 3
<code>Ulaz: 123 10</code>	<code>Ulaz: 3251 0</code>	<code>Ulaz: 12541 1024</code>
<code>Izlaz: 4294967285</code>	<code>Izlaz: 4294967295</code>	<code>Izlaz: 4294966271</code>

**Zadatak 1.12** Napisati funkciju koja računa koliko petica bi imao ceo neoznačen broj  $x$  u oktalnom zapisu. Napisati program koji tu funkciju testira za broj koji se zadaje sa standardnog ulaza.

Test 1	Test 2	Test 3
<code>Ulaz: 123</code>	<code>Ulaz: 3245</code>	<code>Ulaz: 100328</code>
<code>Izlaz: 0</code>	<code>Izlaz: 2</code>	<code>Izlaz: 1</code>

Milena: Naredne zadatke prebaciti da budu nakon rekurzije.

**Zadatak 1.13** Napisati rekurzivnu funkciju vraća broj bitova koji su postavljeni na 1, u binarnoj reprezentaciji njenog celobrojnog argumenta. Napisati program koji testira napisanu funkciju za brojeve koji se učitavaju sa standardnog ulaza zadati u heksadekadnom formatu.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
Ulaz: 0x7F    Izlaz: 7	Ulaz: 0x80    Izlaz: 1	Ulaz: 0x00FF00FF    Izlaz: 16
		<i>Test 4</i>
		Ulaz: 0xFFFFFFFF    Izlaz: 32

### Zadatak 1.14

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

[illegible]

**Zadatak 1.15** Nina: Ovo je prebaceno iz poglavlja sa pokazivacima. Napisati rekurzivnu funkciju za određivanje najveće cifre u oktalnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. Uputstvo: binarne cifre grupisati u podgrupe od po tri cifre, počev od bitova najmanje težine.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
Ulaz: 5    Izlaz: 5	Ulaz: 125    Izlaz: 7	Ulaz: 8    Izlaz: 1
<i>Test 4</i>		
Ulaz: 10    Izlaz: 2		

**Zadatak 1.16** Nina: Ovo je prebaceno iz poglavlja sa pokazivacima. Napisati rekurzivnu funkciju za određivanje (dekadne vrednosti) najveće cifre u heksadekadmnom zapisu neoznačenog celog broja korišćenjem bitskih operatora. Uputstvo: binarne cifre grupisati u podgrupe od po četiri cifre, počev od bitova najmanje težine.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
Ulaz: 5	Ulaz: 16	Ulaz: 18
Izlaz: 5	Izlaz: 1	Izlaz: 2
<i>Test 4</i>		
Ulaz: 165		
Izlaz: 10		

## 1.3 Rekurzija

**Zadatak 1.17** Napisati rekurzivnu funkciju koja sumira elemente niza celih brojeva. Napisati program koji testira napisane funkcije. 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.

**Zadatak 1.18** Napisati rekurzivnu funkciju koja izračunava  $x^k$ , za dati realni broj  $x$  i prirodan broj  $k$ . Napisati program koji testira napisanu funkciju za vrednosti koje se unose sa standardnog ulaza.

**Zadatak 1.19** Koristeći uzajamnu (posrednu) rekurziju napisati naredne dve funkcije:

- funkciju **paran** koja proverava da li je broj cifara nekog broja paran i vraća 1 ako jeste, a 0 inače;
- i funkciju **neparan** koja vraća 1 ukoliko je broj cifara nekog broja neparan, a 0 inače.

Napisati program koji testira napisanu funkciju tako što se za heksadekadnu vrednost koja se unosi sa standardnog ulaza ispisuje da li je paran ili neparan.

**Zadatak 1.20** Napisati repno-rekurzivnu funkciju koja izračunava faktorijel broja  $n$ . Napisati program koji testira napisanu funkciju za poizvoljan broj  $n$  ( $n \leq 12$ ) unet sa standardnog ulaza.

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

$$F(0) = 0$$

$$F(1) = 1$$

$$F(n) = a * F(n - 1) + b * F(n - 2)$$

- Napisati rekurzivnu funkciju koja računa  $n$ -ti element u nizu  $F$ .
- Napisati rekurzivnu funkciju koja računa  $n$ -ti element u nizu  $F$  ali tako da se problemi manje dimenzije rešavaju samo jedan put.

Napisati program koji testira napisane funkcije za poizvoljan broj  $n$  ( $n \in \mathbb{N}$ ) unet sa standardnog ulaza.

**Zadatak 1.22** Napisati rekurzivnu funkciju koja prikazuje sve permutacije skupa  $\{1, 2, \dots, n\}$ . Napisati program koji testira napisanu funkciju za poizvoljan prirodan broj  $n$  ( $n \leq 50$ ) unet sa standardnog ulaza.

**Zadatak 1.23** Paskalov trougao se dobija tako što mu je svako polje (izuzev jedinica po krajevima) zbir jednog polja levo i jednog polja iznad.

## 1 Uvodni zadaci

---

```
      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
```

- (a) Napisati rekurzivnu funkciju koja izračunava  $d_n$  kao sumu elemenata  $n$ -te hipotenuze Paskalovog trougla.
- (b) Napisati rekurzivnu funkciju koja izračunava vrednost polja  $(i, j)$ . **Milena:** dodati sta je  $i$  a sta  $j$  tj odakle se broji

**Zadatak 1.24** Napisati rekurzivnu funkciju koja određuje maksimum niza celih brojeva. Napisati program koji testira ovu funkciju, za niz koji se unosi sa standardnog ulaza. Niz neće imati više od 256 elemenata, i njegovi elementi se unose sve do kraja ulaza.

<i>Test 1</i>	<i>Test 2</i>
<pre>   Ulaz:  3 2 1 4 21    Izlaz: 21</pre>	<pre>   Ulaz:  2 -1 0 -5 -10    Izlaz:  2</pre>

  

<i>Test 3</i>	<i>Test 4</i>
<pre>   Ulaz:  1 11 3 5 8 1    Izlaz: 11</pre>	<pre>   Ulaz:  5    Izlaz:  5</pre>

**Zadatak 1.25** Napisati rekurzivnu funkciju koja izračunava skalarni proizvod dva data vektora. Napisati program koji testira ovu funkciju, za nizove koji se unose sa standardnog ulaza. Nizovi neće imati više od 256 elemenata. Prvo se unosi dimenzija nizova, a zatim i sami njihovi elementi.

<i>Test 1</i>	<i>Test 2</i>
<pre>   Ulaz:  3 1 2 3 1 2 3    Izlaz: 14</pre>	<pre>   Ulaz:  2 3 5 2 6    Izlaz: 36</pre>

  

<i>Test 3</i>
<pre>   Ulaz:  0    Izlaz:  0</pre>

**Zadatak 1.26** Napisati rekurzivnu funkciju koja sabira dekadne cifre datog celog broja  $x$ . Napisati program koji testira ovu funkciju, za broj koji se unosi sa standardnog ulaza.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<b>Ulaz:</b> 123    <b>Izlaz:</b> 6	<b>Ulaz:</b> 23156    <b>Izlaz:</b> 17	<b>Ulaz:</b> 1432    <b>Izlaz:</b> 10
<i>Test 4</i>	<i>Test 5</i>	
<b>Ulaz:</b> 1    <b>Izlaz:</b> 1	<b>Ulaz:</b> 0    <b>Izlaz:</b> 0	

**Zadatak 1.27** Napisati rekurzivnu funkciju koja računa broj pojavljivanja elementa  $x$  u nizu  $a$  dužine  $n$ . Napisati program koji testira ovu funkciju, za  $x$  i niz koji se unose sa standardnog ulaza. Niz neće imati više od 256 elemenata. Prvo se unosi  $x$ , a zatim elementi niza sve do kraja ulaza.

<i>Test 1</i>	<i>Test 2</i>
<b>Ulaz:</b> 4 1 2 3 4    <b>Izlaz:</b> 1	<b>Ulaz:</b> 11 3 2 11 14 11 43 1    <b>Izlaz:</b> 2
<i>Test 3</i>	
<b>Ulaz:</b> 1 3 21 5 6    <b>Izlaz:</b> 0	

**Zadatak 1.28** Napisati rekurzivnu funkciju koja ispituje da li je data niska palindrom. Napisati program koji testira ovu funkciju. Pretpostaviti da niska neće imati više od 31 karaktera, i da se unosi sa standardnog ulaza.

<i>Test 1</i>		<i>Test 2</i>			
Ulaz:	programiranje	Ulaz:	anavolimilovana		
Izlaz:	ne	Izlaz:	da		
<i>Test 3</i>		<i>Test 4</i>		<i>Test</i>	
Ulaz:	a	Ulaz:	aba	Ulaz:	aa
Izlaz:	da	Izlaz:	da	Izlaz:	da

**Zadatak 1.29** Napisati rekurzivnu funkciju kojom se proverava da li su tri zadata broja uzastopni članovi niza. Potom, napisati program koji je testira. Sa standardnog ulaza se unose najpre tri tražena broja, a zatim elementi niza, sve do kraja ulaza. Pretpostaviti da neće biti uneto više od 256 brojeva.

<i>Test 1</i>	<i>Test 2</i>
<b>Ulaz:</b> 1 2 3 4 1 2 3 4 5    <b>Izlaz:</b> da	<b>Ulaz:</b> 1 2 3 11 1 2 4 3 6    <b>Izlaz:</b> ne

### Test 3

```
|| Ulaz:      1 2 3 1 2
|| Izlaz:     ne
```

**Zadatak 1.30** Napisati rekurzivnu funkciju koja prikazuje sve varijacije sa ponavljanjem dužine  $n$  skupa  $\{a, b\}$ , i program koji je testira, za  $n$  koje se unosi sa standardnog ulaza.

### Test 1

```
|| Ulaz:      3
|| Izlaz:     a a a
||             a a b
||             a b a
||             a b b
||             b a a
||             b a b
||             b b a
||             b b b
```

**Zadatak 1.31** *Hanojske kule*: Data su tri vertikalna štapa, na jednom se nalazi  $n$  diskova poluprečnika 1,2,3,... do  $n$ , tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala dva štapa su prazna. Potrebno je premestiti diskove na drugi štap tako da budu u istom redosledu, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg, a preostali štap se koristi kao pomoćni štap prilikom premeštanja. Napisati program koji za proizvoljnu vrednost  $n$ , koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

**Zadatak 1.32** *Modifikacija Hanojskih kula*: Data su četiri vertikalna štapa, na jednom se nalazi  $n$  diskova poluprečnika 1,2,3,... do  $n$ , tako da se najveći nalazi na dnu, a najmanji na vrhu. Ostala tri štapa su prazna. Potrebno je premestiti diskove na drugi štap tako da budu u istom redosledu, premestajući jedan po jedan disk, pri čemu se ni u jednom trenutku ne sme staviti veći disk preko manjeg, pri čemu se preostala dva štapa koriste kao pomoćni štapovi prilikom premeštanja. Napisati program koji za proizvoljnu vrednost  $n$ , koja se unosi sa standardnog ulaza, prikazuje proces premeštanja diskova.

## 1.4 Rešenja

Rešenje [1.1](#)

Rešenje [1.2](#)

Rešenje [1.3](#)



Rešenje 1.4

Rešenje 1.5

Rešenje 1.6

Rešenje 1.7

Rešenje 1.8

Rešenje 1.9

Rešenje 1.10

Rešenje 1.11

Rešenje 1.12

Rešenje 1.13

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n"); /* Da li komentari rade ččžš*/
    return 0;
6 }
```

Rešenje 1.15

Rešenje 1.16

Rešenje 1.17

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n"); /* Da li komentari rade ččžš*/
    return 0;
6 }
```

Rešenje 1.18

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n"); /* Da li komentari rade ččžš*/
    return 0;
6 }
```

Rešenje [1.19](#)

Rešenje [1.20](#)

Rešenje [1.21](#)

Rešenje [1.22](#)

Rešenje [1.23](#)

Rešenje [1.24](#)

Rešenje [1.25](#)

Rešenje [1.26](#)

Rešenje [1.27](#)

Rešenje [1.28](#)

Rešenje [1.29](#)

# Glava 2

## Pokazivači

### 2.1 Pokazivačka aritmetika

**Zadatak 2.1** Milen: ovako definisan zadatak zahteva dva programa kao resenja, a ne jedan sa definisane dve funkcije. Za dati celobrojni niz dimenzije  $n$ , napisati funkciju koja obrće njegove elemente:

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

Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju niza  $n$  ( $0 < n \leq 100$ ), a zatim elemente niza. Prikazati sadržaj niza posle poziva funkcije za obrtanje elemenata niza.

<i>Test 1</i>	<i>Test 2</i>
<pre>Ulaz:  3       1 -2 3 Izlaz: 3 -2 1</pre>	<pre>Ulaz:  0 Izlaz: Greska: neodgovarajuca dimenzija niza.</pre>

**Zadatak 2.2** 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 učitanoog niza.

### Test 1

```
|| Ulaz: 3
||      -1.1 2.2 3.3
|| Izlaz: zbir = 4.400
||         proizvod = -7.986
||         min = -1.100
||         max = 3.300
```

**Zadatak 2.3** Korišćenjem pokazivačke sintakse, napisati funkciju koja vrednosti elemenata u prvoj polovini niza povećava za jedan, a u drugoj polovini smanjuje za jedan. Ukoliko niz ima neparan broj elemenata, onda vrednost srednjeg elementa niza ostaviti nepromenjenim. Napisati program koji testira napisanu funkciju. Sa standardnog ulaza učitati dimenziju  $n$  ( $0 < n \leq 100$ ) celobrojong niza, a zatim i elemente niza. Na standardni izlaz ispisati rezultat primene napisane funkcije nad učitanim nizom. *Jelena: Sta kazete na to da prekoracenja dimenzije niza u razlicitim zadacima razlicito obradjujemo. Na primer, mozemo da unosimo dimenziju niza sve dok se ne unese broj koji je u odgovarajucem opsegu, ili mozemo da dimenziju postavimo na 1 ako je korisnik uneo broj manji od 1, a na MAX ako je korisnik uneo broj veci od MAX, itd?*

### Test 1

```
|| Ulaz: 5
||      1 2 3 4 5
|| Izlaz: 2 3 3 3 4
```

### Test 2

```
|| Ulaz: 4
||      4 -3 2 -1
|| Izlaz: 5 -2 1 -2
```

### Test 3

```
|| Ulaz: 0
|| Izlaz: Greska: neodgovarajuca dimenzija niza.
```

### Test 4

```
|| Ulaz: 101
|| Izlaz: Greska: neodgovarajuca dimenzija niza.
```

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

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

*Jelena: Da li je ok da ovaj zadatak pod a i b resim na nacin na koji sam resila, odnosno, da jedno od ta dva resenja iskomentarisem? Milena: Meni se cini da je bolje bez komentarisanja, vec da su oba prisutna.*

<i>Test 1</i>	<i>Test 2</i>
<pre> Poziv: ./a.out prvi 2. treci -4 Izlaz: 5       0 ./a.out       1 prvi       2 2.       3 treci       4 -4       . p 2 - </pre>	<pre> Poziv: ./a.out Izlaz: 1       0 ./a.out       . </pre>

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

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> Poziv: ./a.out programiranje anavolimilovana topot ana anagram t Izlaz: 4 </pre>	<pre> Poziv: ./a.out a b 11 212 Izlaz: 4 </pre>	<pre> Poziv: ./a.out Izlaz: 0 </pre>

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

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre> Poziv: ./a.out ulaz.txt 1 ulaz.txt: Ovo je sadržaj datoteke i u njoj ima reci koje imaju           1 karakter Izlaz: 3 </pre>	<pre> Poziv: ./a.out ulaz.txt Izlaz: Greska: Nedovoljan broj argumenata komandne linije.         Program se poziva sa ./a.out ime_dat br_karaktera. </pre>	<pre> Poziv: ./a.out ulaz.txt 2 (ne postoji datoteka ulaz.txt) Izlaz: Greska: Neuspesno otvaranje datoteke ulaz.txt. </pre>

**Zadatak 2.7** Napisati program koji kao prvi argument komandne linije prihvata

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

### Test 1

```
Poziv: ./a.out ulaz.txt ke -s
ulaz.txt: Ovo je sadrzaj datoteke i u njoj ima reci koje se
          završavaju na ke
Izlaz: 2
```

### Test 2

```
Poziv: ./a.out ulaz.txt sa -p
ulaz.txt: Ovo je sadrzaj datoteke i u njoj ima reci koje
          pocijnu sa sa
Izlaz: 3
```

### Test 3

```
Poziv: ./a.out ulaz.txt sa -p
(ne postoji datoteka ulaz.txt)
Izlaz: Greska: Neuspesno otvaranje datoteke ulaz.txt.
```

### Test 3

```
Poziv: ./a.out ulaz.txt
Izlaz: Greska: Nedovoljan broj argumenata komandne linije.
       Program se poziva sa ./a.out ime_dat suf/pref -s/-p.
```

## 2.2 Višedimenzioni nizovi

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

- (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 dimenziju 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 učitanе matrice.

## Test 1

```

Ulaz:  3 1 -2 3 4 -5 6 7 -8 9
Izlaz: 1 -2 3
        4 -5 6
        7 -8 9
        trag = 5
        euklidska norma = 16.88
        vandijagonalna norma = 11

```

## Test 2

```

Ulaz:  0
Izlaz: Greska: neodgovarajuca dimenzija matrice.

```

**Zadatak 2.9** 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 dimenziju kvadratnih matrica  $n$  ( $0 < n \leq 100$ ), a zatim i elemente matrica. Na standardni izlaz ispisati „da“ ako su matrice jednake, „ne“ ako nisu a zatim ispisati zbir i proizvod učitanih matrica.

## Test 1

```

Ulaz:  3
        1 2 3 1 2 3 1 2 3
        1 2 3 1 2 3 1 2 3
Izlaz:  da
        Zbir matrica je:
        2 4 6
        2 4 6
        2 4 6
        Proizvod matrica je:
        6 12 18
        6 12 18
        6 12 18

```

**Zadatak 2.10** 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.

### Test 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
Izlaz:    Refleksivnost: ne
          Simetricnost: ne
          Tranzitivnost: da
          Refleksivno zatvorenje:
          1 0 0 0
          0 1 1 0
          0 0 1 0
          0 0 0 1
          Simetricno zatvorenje:
          1 0 0 0
          0 1 1 0
          0 1 1 0
          0 0 0 0
          Refleksivno-tranzitivno zatvorenje:
          1 0 0 0
          0 1 1 0
          0 0 1 0
          0 0 0 0
```

**Zadatak 2.11** 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 najveći element matrice na sporednoj dijagonali, indeks kolone koja sadrži najmanji element, indeks vrste koja sadrži najveći element i broj negativnih elemenata učitane matrice.

<i>Test 1</i>	<i>Test 2</i>
<pre> Poziv: ./a.out 3 Ulaz:  1 2 3         -4 -5 -6         7 8 9 Izlaz:  7 2 2 3           </pre>	<pre> Poziv: ./a.out 4 Ulaz:  -1 -2 -3 -4         -5 -6 -7 -8         -9 -10 -11 -12         -13 -14 -15 -16 Izlaz:  -4 3 0 16           </pre>
<i>Test 3</i>	
<pre> Poziv: ./a.out Izlaz: Greska: Nedovoljan broj argumenata komandne linije.         Program se poziva sa ./a.out dim_matrice.           </pre>	

**Zadatak 2.12** 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 učitanoj matrici.

<i>Test 1</i>	<i>Test 2</i>
<pre> Ulaz:  4         1 0 0 0         0 1 0 0         0 0 1 0         0 0 0 1 Izlaz: da           </pre>	<pre> Ulaz:  3         1 2 3         5 6 7         1 4 2 Izlaz: ne           </pre>
<i>Test 3</i>	
<pre> Ulaz:  33 Izlaz: Greska: neodgovarajuca dimenzija matrice.           </pre>	

**Zadatak 2.13** 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  $m$  ( $0 < m \leq 10$ ), a zatim i elemente matrice

## 2 Pokazivači

---

(pozivom gore napisane funkcije). Na standardni izlaz spiralno ispisati elemente učitane matrice.

<i>Test 1</i>	<i>Test 2</i>
<pre>   Ulaz:  3 3          1 2 3          4 5 6          7 8 9    Izlaz: 1 2 3 6 9 8 7 4 5</pre>	<pre>   Ulaz:  3 4          1 2 3 4          5 6 7 8          9 10 11 12    Izlaz: 1 2 3 4 8 12 11 10 9 5 6 7</pre>
<i>Test 3</i>	
<pre>   Ulaz:  11 4    Izlaz: Greska: neodgovarajuće dimenzije matrice.</pre>	

**Zadatak 2.14** 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.

*Test 1*

```
|| Ulaz:  3
||       1 2 3
||       4 5 6
||       7 8 9
||       8
|| Izlaz: 510008400 626654232 743300064
||       1154967822 1419124617 1683281412
||       1799927244 2211595002 2623262760
```

## 2.3 Dinamička alokacija memorije

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

<i>Test 1</i>	<i>Test 2</i>
<pre>   Ulaz:  3          1 -2 3    Izlaz: 3 -2 1</pre>	<pre>   Ulaz:  -1    Izlaz: malloc(): neuspela alokacija memorije.</pre>

**Zadatak 2.16** Napisati program koji sa standardnog ulaza učitava niz celih brojeva. Brojevi se unose sve dok se ne unese nula. Ne praviti nikakve pretpostavke

o dimenziji niza. Na standardni izlaz ispisati ovaj niz brojeva u obrnutom poretku. Zadatak uraditi na dva načina:

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

<i>Test 1</i>	<i>Test 2</i>
<pre>   Ulaz:  1 -2 3 -4 0    Izlaz: -4 3 -2 1</pre>	<pre>   Ulaz:  0    Izlaz:</pre>

**Zadatak 2.17** Napisati funkciju koja kao rezultat vraća nisku koja se dobija nadovezivanjem dve niske, bez promene njihovog sadržaja. Napisati program koji testira rad napisane funkcije. Sa standardnog ulaza učitati dve niske karaktera (pretpostaviti da niske nisu duže od 1000 karaktera i da ne sadrže praznine). Na standardni izlaz ispisati nisku koja se dobija njihovim nadovezivanjem. Za rezultujuću nisku dinamički alocirati memoriju.

<i>Test 1</i>
<pre>   Ulaz:  Jedan Dva    Izlaz: JedanDva</pre>

**Zadatak 2.18** Napisati program koji sa standardnog ulaza učitava matricu celih 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.

<i>Test 1</i>
<pre>   Ulaz:  2 3            1.2 2.3 3.4            4.5 5.6 6.7    Izlaz: 6.80</pre>

**Zadatak 2.19** Data je celobrojna matrica dimenzije  $n \times m$  napisati:

- (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.

### Test 1

```
Ulaz:  2 3
        1 -2 3
        -4 5 -6
Izlaz:  1
        -4 5
```

**Zadatak 2.20** 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.

### Test 1

```
Ulaz:  Unesite dimenzije matrice:
        2 3
        Unesite elemente matrice:
        1 2 3
        4 5 6
Izlaz:  Kolona pod rednim brojem 3 ima najveći zbir.
```

**Zadatak 2.21** Data je kvadratna realna matrica dimenzije  $n$ .

- Napisati funkciju koja izračunava zbir apsolutnih vrednosti matrice ispod sporedne dijagonale.
- 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.

### Test 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
Izlaz:  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 2.22** 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 pronade ime reprezentacije čijih sličica ima najmanje. Dobijene podatke ispisati na standardni izlaz. Napomena: za realokaciju memorije koristiti `realloc()` funkciju. **Jelena: treba dodati test primer.**

**Zadatak 2.23** 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. **Jelena: treba dodati test primer.**

**Zadatak 2.24** Napisati program koji na osnovu dve 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. **Jelena: treba dodati test primer.**

**Zadatak 2.25** 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. Sa standardnog ulaza se prvo unosi dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz. **Jelena: treba dodati test primer.**

## 2.4 Pokazivači na funkcije

**Zadatak 2.26** 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`).

## Test 1

```
Poziv: ./a.out sin
Ulaz: Unesite krajeve intervala:
      -0.5 1
      Koliko tacaka ima na ekvidistantnoj mrezi (ukljucujuci krajeve
      intervala)?
      4
Izlaz:      x      sin(x)
      -----
      | -0.50000 | -0.47943 |
      |  0.00000 |  0.00000 |
      |  0.50000 |  0.47943 |
      |  1.00000 |  0.84147 |
      -----
```

## Test 2

```
Poziv: ./a.out cos
Ulaz: Unesite krajeve intervala:
      0 2
      Koliko tacaka ima na ekvidistantnoj mrezi (ukljucujuci krajeve
      intervala)?
      4
Izlaz:      x      cos(x)
      -----
      |  0.00000 |  1.00000 |
      |  0.66667 |  0.78589 |
      |  1.33333 |  0.23524 |
      |  2.00000 | -0.41615 |
      -----
```

**Zadatak 2.27** 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):

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

## Test 1

```
Ulaz:  tan 1.570795 10000
Izlaz: -10134.5
```

## Test 2

```
Ulaz:  log 0 1000000
Izlaz: -13.81551
```

**Zadatak 2.28** Napisati funkciju koja određuje integral funkcije  $f(x)$  na intervalu  $[a, b]$ . Adresa funkcije  $f$  se prenosi kao parametar. Integral se računa prema formuli:

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

Vrednost  $h$  se izračunava po formuli  $h = (b - a)/n$ , dok se vrednosti  $n$ ,  $a$  i  $b$  unose

sa standardnog ulaza kao i ime funkcije iz zaglavlja `math.h`. Na standardni izlaz ispisati vrednost integrala. **Jelena: treba dodati test primer.**

**Zadatak 2.29** 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 pretrage i  $n$ , a na standardni izlaz ispisuje vrednost odgovarajućeg integrala. **Jelena: treba dodati test primer.**

## 2.5 Rešenja

### Rešenje 2.1

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX 100
5
6 /* Funkcija obrne elemente niza koriscenjem indekse sintakse */
7 void obrni_niz_v1(int a[] , int n)
8 {
9     int i, j;
10
11     for(i = 0, j = n-1; i < j; i++, j--) {
12         int t = a[i];
13         a[i] = a[j];
14         a[j] = t;
15     }
16 }
17
18 /* Funkcija obrne elemente niza koriscenjem pokazivacke
19    sintakse. Umesto "void obrni_niz(int *a, int n)" potpis
20    metode bi mogao da bude i "void obrni_niz(int a[], int n)".
21    U oba slucaja se argument funkcije "a" tumaci kao pokazivac,
22    ili tacnije, kao adresa prvog elementa niza. U odnosu na
23    njega se odredjuju adrese ostalih elemenata u nizu */
24 void obrni_niz_v2(int *a, int n)
25 {
26     /* Pokazivaci na elemente niza a */
27     int *prvi, *poslednji;
28
29     for(prvi = a, poslednji = a + n - 1;
30         prvi < poslednji; prvi++, poslednji--) {
31         int t = *prvi;

```

```

    *prvi = *poslednji;
34    *poslednji = t;
    }
36 }

38 /* Funkcija obrće elemente niza koriscenjem pokazivacke
    sintakse - modifikovano koriscenje pokazivaca */
40 void obrni_niz_v3(int *a, int n)
{
42     /* Pokazivaci na elemente niza a */
    int *prvi, *poslednji;
44
    /* Obrćemo niz */
46     for(prvi = a, poslednji = a + n - 1; prvi < poslednji; ) {
        int t = *prvi;
48
        /* Na adresu na koju pokazuje pokazivac "prvi" postavlja se
50         vrednost koja se nalazi na adresi na koju pokazuje
        pokazivac "poslednji". Nakon toga se pokazivac "prvi"
52         uvecava za jedan sto za posledicu ima da "prvi" pokazuje
        na sledeci element u nizu */
54         *prvi++ = *poslednji;

56         /* Vrednost promenljive "t" se postavlja na adresu na koju
        pokazuje pokazivac "poslednji". Ovaj pokazivac se zatim
58         umanjuje za jedan, sto za posledicu ima da pokazivac
        "poslednji" sada pokazuje na element koji mu prethodi u
60         nizu */
        *poslednji-- = t;
62     }
    }
64

66 int main()
{
68     /* Deklaracija niza a od najvise MAX elemenata */
    int a[MAX];

70     /* Broj elemenata niza a */
    int n;
72
74     /* Pokazivac na elemente niza a */
    int *p;

76     /* Unosimo dimenziju niza */
    scanf("%d", &n);
78
80     /* Proveravamo da li je prekoraceno ogranicenje dimenzije */
    if(n <= 0 || n > MAX) {
        fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
82         exit(EXIT_FAILURE);
    }
84
    /* Unosimo elemente niza */
86     for(p = a; p - a < n; p++)
        scanf("%d", p);
88
```



```

    obrni_niz_v1(a,n);
90 // obrni_niz_v2(a,n);
    // obrni_niz_v3(a,n);

92
    /* Prikazujemo sadrzaj niza nakon obrtanja */
94 for(p = a; p - a < n; p++)
    printf("%d ", *p);
96 printf("\n");

98 return 0;
}

```

## Rešenje 2.2

```

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

4 #define MAX 100

6 /* Funkcija racuna 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 racuna 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 racuna najmanji element niza */
double min(double *a, int n)
30 {
    /* Za najmanji element se najpre postavlja prvi element */
32    double min = a[0];
    int i;

34
    /* Ispitujemo da li se medju ostalim elementima niza
36     nalazi najmanji */
    for(i = 1; i < n; i++)
38         if ( a[i] < min )
            min = a[i];

40
    return min;
}

```

```
42 }
44 /* Funkcija racuna najveći element niza */
double max(double *a, int n)
46 {
    /* Za najveći element se najpre postavlja prvi element */
48     double max = *a;

50     /* Ispitujemo da li se medju ostalim elementima niza
        nalazi najveći */
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;

64     /* Ucitavamo dimenziju niza */
66     scanf("%d", &n);

68     /* Proveravamo da li je prekoraceno ogranicenje dimenzije */
    if(n <= 0 || n > MAX) {
70         fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
        exit(EXIT_FAILURE);
72     }

74     /* Unosimo elemente niza */
    for(i = 0; i < n; i++)
76         scanf("%lf", a + i);

78     /* Testiramo definisane funkcije */
    printf("zbir = %5.3f\n", zbir(a, n));
    printf("proizvod = %5.3f\n", proizvod(a, n));
    printf("min = %5.3f\n", min(a, n));
    printf("max = %5.3f\n", max(a, n));
82

84     return 0;
}
```

### Rešenje 2.3

```
#include <stdio.h>
2 #include <stdlib.h>
#define MAX 100
4
/* Funkcija povecava za jedan sve elemente u prvoj polovini niza
6  a smanjuje za jedan sve elemente u drugoj polovini niza.
    Ukoliko niz ima neparan broj elemenata, srednji element
8  ostaje nepromenjen */
```

```
void povecaj_smanji (int *a , int n)
10 {
    int *prvi = a;
12     int *poslednji = a+n-1;

14     while( prvi < poslednji ){

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

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

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

26         /* Pokazivac poslednji se pomera na prethodni element */
28         poslednji--;
           }
30 }

32 void povecaj_smanji_sazetije(int *a , int n)
{
34     int *prvi = a;
    int *poslednji = a+n-1;

36     while( prvi < poslednji ){
38         (*prvi++)++;
        (*poslednji--)--;
40     }
}

42 int main()
44 {
    int a[MAX];
46     int n;
    int *p;

48     /* Unosimo broj elemenata */
50     scanf("%d", &n);

52     /* Proveravamo da li je prekoraceno ogranicenje dimenzije */
    if(n <= 0 || n > MAX) {
54         fprintf(stderr, "Greska: neodgovarajuca dimenzija niza.\n");
        exit(EXIT_FAILURE);
56     }

58     /* Unosimo elemente niza */
    for(p = a; p - a < n; p++)
60         scanf("%d", p);

62     povecaj_smanji(a,n);
    /* povecaj_smanji_sazetije(a,n); */
64 }
```

```
/* Prikaz niza nakon modifikacije */
66 for(p = a; p - a < n; p++)
    printf("%d ", *p);
68 printf("\n");

70 return 0;
}
```

### Rešenje 2.4

```
#include <stdio.h>

2
/* Argumenti funkcije main mogu da budu broj argumenta komandne
4 linije (int argc) i niz argumenata komandne linije
   (niz niski) (char *argv[] <=> char** argv) */
6 int main(int argc, char *argv[])
{
8     int i;

10    /* Ispisujemo broj argumenata komandne linije */
    printf("%d\n", argc);

12
    /* Ispisujemo argumente komandne linije */
    /* koristeći indeksnu sintaksu */
14    for(i=0; i<argc; i++) {
16        printf("%d %s\n", i, argv[i]);
    }

18
    /* koristeći pokazivačku sintaksu */
20    i=argc;
    for (; argc>0; argc--)
22        printf("%d %s\n", i-argc, *argv++);

24
    /* Nakon ove petlje "argc" će biti jednako nuli a "argv" će
26    pokazivati na polje u memoriji koje se nalazi iza
    poslednjeg argumenta komandne linije. Kako smo u
28    promenljivoj "i" sacuvali vrednost broja argumenta
    komandne linije to sada možemo ponovo da postavimo
30    "argv" da pokazuje na nulti argument komandne linije */
    argv = argv - i;
32    argc = i;

34    /* Ispisujemo 0-ti karakter svakog od argumenata komandne linije
    */

36    /* koristeći indeksnu sintaksu */
    for(i=0; i<argc; i++)
38        printf("%c ", argv[i][0]);
    printf("\n");

40
    /* koristeći pokazivačku sintaksu */

42
    for (i=0 ; i<argc; i++ )
44        printf("%c ", **argv++);
```

```
46     return 0;
    }
```

### Rešenje 2.5

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

    /* Multi argument komandne linije je ime izvrsnog programa */
    for(i = 1; i < argc; i++)
        if(palindrom(*(argv+i)))
            n++;

    printf("%d\n", n);
    return 0;
}
```

### Rešenje 2.6

```
1  #include<stdio.h>
   #include<stdlib.h>
3
   #define MAX_KARAKTERA 100
5
   /* Funkcija strlen() iz standardne biblioteke */
7  int duzina(char *s)
   {
9      int i;
      for(i = 0; *(s+i); i++)
11         ;
      return i;
13 }

15 int main(int argc, char **argv)
   {
17     char rec[MAX_KARAKTERA];
      int br = 0, i = 0, n;
19     FILE *in;
```

```
21  /* Ako korisnik nije uneo trazene argumente,
    prijavljujemo gresku */
23  if(argc < 3) {
    printf("Greska: ");
25  printf("Nedovoljan broj argumenata komandne linije.\n");
    printf("Program se poziva sa %s ime_dat br_karaktera.\n",
27                                     argv[0]);

    exit(EXIT_FAILURE);
29  }

31  /* Otvaramo datoteku sa imenom koje se zadaje kao prvi
    argument komandne linije. */
33  in = fopen(*(argv+1), "r");
    if(in == NULL){
35      fprintf(stderr, "Greska: ");
      fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
37                                     argv[1]);

      exit(EXIT_FAILURE);
39  }

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

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

49  printf("%d\n", br);

51  /* Zatvaramo datoteku */
    fclose(in);
53  return 0;
}
```

### Rešenje 2.7

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

3

    #define MAX_KARAKTERA 100

5

    /* Funkcija strcpy() iz standardne biblioteke */
7  void kopiranje_niske(char *dest, char *src)
    {
9      int i;
      for (i = 0; *(src+i); i++)
11         *(dest+i) = *(src+i);
    }

13

    /* Funkcija strcmp() iz standardne biblioteke */
15  int poredjenje_niski(char *s, char *t)
    {
17      int i;
```

```

19     for (i = 0; *(s+i) == *(t+i); i++)
        if(*(s+i) == '\0')
            return 0;
21     return *(s+i) - *(t+i);
}

23
/* Funkcija strlen() iz standardne biblioteke */
25 int duzina_niske(char *s)
{
27     int i;
    for(i = 0; *(s+i); i++)
29         ;
    return i;
31 }

33 /* Funkcija ispituje da li je niska zadata drugim argumentom
    funkcije sufiks niske zadate prvi argumentom funkcije */
35 int sufiks_niske(char *niska, char *sufiks) {
    if(duzina_niske(sufiks) <= duzina_niske(niska) &&
37         poredjenje_niski(niska + duzina_niske(niska) -
            duzina_niske(sufiks), sufiks) == 0)
39         return 1;
    return 0;
41 }

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

55
int main(int argc, char **argv)
57 {
    /* Ako korisnik nije uneo trazene argumente,
59     prijavljujemo gresku */
    if(argc < 4) {
61         printf("Greska: ");
        printf("Nedovoljan broj argumenata komandne linije.\n");
63         printf("Program se poziva sa %s ime_dat suf/pref -s/-p.\n",
            argv[0]);
65         exit(EXIT_FAILURE);
    }

67
    FILE *in;
69     int br = 0, i = 0, n;
    char rec[MAX_KARAKTERA];

71
    in = fopen(*(argv+1), "r");
73     if(in == NULL) {

```

```
75     fprintf(stderr, "Greska: ");
76     fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
77               argv[1]);
78     exit(EXIT_FAILURE);
79 }
80
81 /* Proveravamo kojom opcijom je pozvan program a zatim
82    učitavamo reci iz datoteke brojimo koliko reci
83    zadovoljava trazeni uslov */
84 if(!(poredjenje_niski(*(argv + 3), "-s")))
85     while(fscanf(in, "%s", rec) != EOF)
86         br += sufiks_niske(rec, *(argv+2));
87 else if (!(poredjenje_niski(*(argv+3), "-p")))
88     while(fscanf(in, "%s", rec) != EOF)
89         br += prefiks_niske(rec, *(argv+2));
90
91 printf("%d\n", br);
92 fclose(in);
93 return 0;
94 }
```

### Rešenje 2.8

```
1 #include <stdio.h>
2 #include <math.h>
3 #include <stdlib.h>
4
5 #define MAX 100
6
7 /* Deklarisemo funkcije koje cemo kasnije da definisemo */
8 double euklidska_norma( int M[][MAX], int n);
9 int trag(int M[][MAX], int n);
10 int gornja_vandijagonalna_norma(int M[][MAX], int n);
11
12 int main()
13 {
14     int A[MAX][MAX];
15     int i,j,n;
16
17     /* Unosimo dimenziju kvadratne matrice */
18     scanf("%d",&n);
19
20     /* Proveravamo da li je prekoraceno ogranicenje */
21     if( n > MAX || n <= 0) {
22         fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
23         fprintf(stderr, "matrice.\n");
24         exit(EXIT_FAILURE);
25     }
26
27     /* Popunjavamo vrstu po vrstu matrice */
28     for(i = 0; i<n; i++)
29         for (j=0 ; j<n; j++)
30             scanf("%d",&A[i][j]);
31
32     /* Ispis elemenata matrice koriscenjem indeksne sintakse.
```



```

    Ispis vrsimo vrstu po vrstu */
34 for(i = 0; i<n; i++) {
    /* Ispisujemo elemente i-te vrste */
36     for ( j=0 ; j<n; j++)
        printf("%d ", A[i][j]);
38     printf("\n");
    }
40
    /* 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
    44 iza poslednjeg elementa prve vrste smesten je prvi element
    druge vrste za kojim slede svi elementi te vrste
    i tako dalje redom */
46
    /*
    48 for( i = 0; i<n; i++) {
        for ( j=0 ; j<n; j++)
            printf("%d ", *(A+i+j));
    52     printf("\n");
    }
    54 */

    int tr = trag(A,n);
    printf("trag = %d\n",tr);
    58
    printf("euklidska norma = %.2f\n",euklidska_norma(A,n));
    60 printf ("vandijagonalna norma = %d\n",
        gornja_vandijagonalna_norma(A,n));
    62
    return 0;
    64 }

    66 /* Definisemo funkcije koju smo ranije deklarirali */

    68 /* Funkcija izracunava trag matrice */
    int trag(int M[][MAX], int n)
    70 {
        int trag = 0,i;
    72     for(i=0; i<n; i++)
        trag += M[i][i];
    74     return trag;
    }
    76

    78 /* Funkcija izracunava euklidsku normu matrice */
    double euklidska_norma(int M[][MAX], int n)
    {
    80     double norma = 0.0;
        int i,j;
    82
        for(i= 0; i<n; i++)
    84             for(j = 0; j<n; j++)
                norma += M[i][j] * M[i][j];
    86
        return sqrt(norma);
    88 }

```

```
90 /* Funkcija izracunava gornju vandijagonalnu normu matrice */
91 int gornja_vandijagonalna_norma(int M[][MAX], int n)
92 {
93     int norma =0;
94     int i,j;
95
96     for(i=0 ;i<n; i++) {
97         for(j = i+1; j<n; j++)
98             norma += abs(M[i][j]);
99     }
100
101     return norma;
102 }
```

### Rešenje 2.9

```
#include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX 100
5
6 /* Funkcija ucitava elemente kvadratne matrice dimenzije n sa
7    standardnog ulaza */
8 void ucitaj_matricu(int m[][MAX], int n)
9 {
10     int i, j;
11
12     for(i=0; i<n; i++)
13         for(j=0; j<n; j++)
14             scanf("%d", &m[i][j]);
15 }
16
17 /* Funkcija ispisuje elemente kvadratne matrice dimenzije n na
18    standardni izlaz */
19 void ispisi_matricu(int m[][MAX], int n) {
20     int i, j;
21
22     for(i=0; i<n; i++) {
23         for(j=0; j<n; j++)
24             printf("%d ", m[i][j]);
25         printf("\n");
26     }
27 }
28
29 /* Funkcija proverava da li su zadate kvadratne matrice a i b
30    dimenzije n jednake */
31 int jednake_matrice(int a[][MAX], int b[][MAX], int n) {
32     int i, j;
33
34     for(i=0; i<n; i++)
35         for(j=0; j<n; j++)
36             /* Nasli smo elemente na istim pozicijama u matricama
37                koji se razlikuju */
38             if(a[i][j]!=b[i][j])
```

```

    return 0;
40
    /* Prošla je provera jednakosti za sve parove elemenata koji
42     su na istim pozicijama sto znaci da su matrice jednake */
    return 1;
44 }

46 /* Funkcija izracunava zbir dve kvadratne matice */
void saberi(int a[][MAX], int b[][MAX], int c[][MAX], int n)
48 {
    int i, j;
50
    for(i=0; i<n; i++)
52     for(j=0; j<n; j++)
        c[i][j] = a[i][j] + b[i][j];
54 }

56 /* Funkcija izracunava proizvod dve kvadratne matice */
void pomnozi(int a[][MAX], int b[][MAX], int c[][MAX], int n)
58 {
    int i, j, k;
60
    for(i=0; i<n; i++)
62     for(j=0; j<n; j++) {
        /* Mnozimo i-tu vrstu prve sa j-tom kolonom druge matrice */
64         c[i][j] = 0;
        for(k=0; k<n; k++)
66             c[i][j] += a[i][k] * b[k][j];
        }
68 }

70 int main()
{
72     /* Matrice ciji se elementi zadaju sa ulaza */
    int a[MAX][MAX], b[MAX][MAX], c[MAX][MAX];
74
    /* Matrice zbira i proizvoda */
76     int zbir[MAX][MAX], proizvod[MAX][MAX];
78
    /* Dimenzija matrica */
    int n;
80     int i, j;

82     /* Ucitavamo dimenziju kvadratnih matrica i proveravamo njenu
        korektnost */
84     scanf("%d", &n);

86     /* Proveravamo da li je prekoraceno ogranicenje */
    if( n > MAX || n <= 0) {
88         fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
        fprintf(stderr, "matrica.\n");
90         exit(EXIT_FAILURE);
    }
92

    /* Ucitavamo matrice */
94     ucitaj_matricu(a, n);

```

```
    ucitaj_matricu(b, n);
96
    /* Izracunavamo zbir i proizvod matrica */
98    saberi(a, b, zbir, n);
    pomnozi(a, b, proizvod, n);
100
    /* Ispisujemo rezultat */
102    if(jednake_matrice(a, b, n) == 1)
        printf("da\n");
104    else
        printf("ne\n");
106
    printf("Zbir matrica je:\n");
108    ispisi_matricu(zbir, n);
110
    printf("Proizvod matrica je:\n");
    ispisi_matricu(proizvod, n);
112
    return 0;
114 }
```

### Rešenje 2.10

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

14     /* Obilazimo glavnu dijagonalu matrice. Za elemente na glavnoj
       dijagonali vazi da je indeks vrste jednak indeksu kolone */
16     for(i=0; i<n; i++) {
        if(m[i][i] != 1)
18             return 0;
    }
20
    return 1;
22 }

24 /* Funkcija odredjuje refleksivno zatvorenje zadate relacije. Ono
   je odredjeno matricom koja sadrzi sve elemente polazne matrice
26   dopunjene jedinicama na glavnoj dijagonali */
void ref_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
28 {
    int i, j;
30
    /* Prepisujemo vrednosti elemenata matrice pocetne matrice */
32     for(i=0; i<n; i++)
```

```

34     for(j=0; j<n; j++)
        zatvorenje[i][j] = m[i][j];

36     /* Postavljamo na glavnoj dijagonali jedinice */
    for(i=0; i<n; i++)
38         zatvorenje[i][i] = 1;
}

40
42     /* Funkcija proverava da li je relacija simetricna. Relacija je
    simetricna ako za svaki par elemenata vazi: ako je element
44     "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 */
46 int simetricnost (int m[][MAX], int n)
{
48     int i, j;

50     /* Obilazimo elemente ispod glavne dijagonale matrice i
    uporedjujemo ih sa njima simetricnim elementima */
52     for(i=0; i<n; i++)
        for(j=0; j<i; j++)
54         if(m[i][j] != m[j][i])
            return 0;

56     return 1;
58 }

60 /* Funkcija odredjuje simetricno zatvorenje zadate relacije. Ono
    je odredjeno matricom koja sadrzi sve elemente polazne matrice
62     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++)
        for(j=0; j<n; j++)
            zatvorenje[i][j] = m[i][j];

72     /* Odredjujemo simetricno zatvorenje matrice */
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
74         if(zatvorenje[i][j] == 1)
            zatvorenje[j][i] = 1;
78 }

80
82     /* 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
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++)
91         for(j=0; j<n; j++)
92             /* Pokusavamo da pronadjemo element koji narusava
93              * tranzitivnost */
94             for(k=0; k<n; k++)
95                 if(m[i][k] == 1 && m[k][j] == 1 && m[i][j] == 0)
96                     return 0;
97
98     return 1;
99 }
100
101 /* Funkcija odredjuje refleksivno-tranzitivno zatvorenje
102    zadate relacije koriscenjem Varsalovog algoritma */
103 void tran_zatvorenje(int m[][MAX], int n, int zatvorenje[][MAX])
104 {
105     int i, j, k;
106
107     /* Kopiramo pocetnu matricu u matricu rezultata */
108     for(i=0; i<n; i++)
109         for(j=0; j<n; j++)
110             zatvorenje[i][j] = m[i][j];
111
112     /* Primenom Varsalovog algoritma odredjujemo
113        refleksivno-tranzitivno zatvorenje matrice */
114     for(k=0; k<n; k++)
115         for(i=0; i<n; i++)
116             for(j=0; j<n; j++)
117                 if((zatvorenje[i][k] == 1) && (zatvorenje[k][j] == 1)
118                    && (zatvorenje[i][j] == 0))
119                     zatvorenje[i][j] = 1;
120 }
121
122 /* Funkcija ispisuje elemente matrice */
123 void pisi_matricu(int m[][MAX], int n)
124 {
125     int i, j;
126
127     for(i=0; i<n; i++) {
128         for(j=0; j<n; j++)
129             printf("%d ", m[i][j]);
130         printf("\n");
131     }
132 }
133
134 int main(int argc, char* argv[])
135 {
136     FILE* ulaz;
137     int m[MAX][MAX];
138     int pomocna[MAX][MAX];
139     int n, i, j, k;
140
141     /* Ako korisnik nije uneo trazene argumente,
142        prijavljujemo gresku */
143     if(argc < 2) {

```

```

146     printf("Greska: ");
147     printf("Nedovoljan broj argumenata komandne linije.\n");
148     printf("Program se poziva sa %s ime_dat.\n", argv[0]);
149     exit(EXIT_FAILURE);
150 }
151
152 /* Otvaramo datoteku za citanje */
153 ulaz = fopen(argv[1], "r");
154 if(ulaz == NULL) {
155     fprintf(stderr, "Greska: ");
156     fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
157             argv[1]);
158     exit(EXIT_FAILURE);
159 }
160
161 /* Ucitavamo dimenziju matrice */
162 fscanf(ulaz, "%d", &n);
163
164 /* Proveravamo da li je prekoraceno ogranicenje */
165 if( n > MAX || n <= 0) {
166     fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
167     fprintf(stderr, "matrice.\n");
168     exit(EXIT_FAILURE);
169 }
170
171 /* Ucitavamo element po element matrice */
172 for(i=0; i<n; i++)
173     for(j=0; j<n; j++)
174         fscanf(ulaz, "%d", &m[i][j]);
175
176 /* Ispisujemo trazene vrednosti */
177 printf("Refleksivnost: %s\n",
178        refleksivnost(m, n) == 1 ? "da" : "ne");
179
180 printf("Simetricnost: %s\n",
181        simetricnost(m, n) == 1 ? "da" : "ne");
182
183 printf("Tranzitivnost: %s\n",
184        tranzitivnost(m, n) == 1 ? "da" : "ne");
185
186 printf("Refleksivno zatvorenje:\n");
187 ref_zatvorenje(m, n, pomocna);
188 pisi_matricu(pomocna, n);
189
190 printf("Simetricno zatvorenje:\n");
191 sim_zatvorenje(m, n, pomocna);
192 pisi_matricu(pomocna, n);
193
194 printf("Refleksivno-tranzitivno zatvorenje:\n");
195 tran_zatvorenje(m, n, pomocna);
196 pisi_matricu(pomocna, n);
197
198 /* Zatvaramo datoteku */
199 fclose(ulaz);
200
201 return 0;

```

```
}
```

### Rešenje 2.11

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX 32
5
6 int max_sporedna_dijagonala(int m[][MAX], int n)
7 {
8     int i, j;
9     /* Trazimo najveći element na sporednoj dijagonali. Za
10        elemente sporedne dijagonale vazi da je zbir indeksa vrste
11        i indeksa kolone jednak n-1. Za početnu vrednost maksimuma
12        uzimamo element u gornjem desnom uglu */
13     int max_na_sporednoj_dijagonali = m[0][n-1];
14     for(i=1; i<n; i++)
15         if(m[i][n-1-i] > max_na_sporednoj_dijagonali)
16             max_na_sporednoj_dijagonali = m[i][n-1-i];
17
18     return max_na_sporednoj_dijagonali;
19 }
20
21 /* Funkcija izracunava indeks kolone najmanjeg elementa */
22 int indeks_min(int m[][MAX], int n)
23 {
24     int i, j;
25     /* Za početnu vrednost minimuma uzimamo element u gornjem
26        levom uglu */
27     int min=m[0][0], indeks_kolone=0;
28
29     for(i=0; i<n; i++)
30         for(j=0; j<n; j++)
31             /* Ako je tekuci element manji od minimalnog */
32             if(m[i][j]<min) {
33                 /* cuvamo njegovu vrednost */
34                 min=m[i][j];
35                 /* i cuvamo indeks kolone u kojoj se nalazi */
36                 indeks_kolone=j;
37             }
38     return indeks_kolone;
39 }
40
41 /* Funkcija izracunava indeks vrste najveceg elementa */
42 int indeks_max(int m[][MAX], int n) {
43     int i, j;
44     /* Za maksimalni element uzimamo gornji levi ugao */
45     int max=m[0][0], indeks_vrste=0;
46
47     for(i=0; i<n; i++)
48         for(j=0; j<n; j++)
49             /* Ako je tekuci element manji od minimalnog */
50             if(m[i][j]>max) {
51                 /* cuvamo njegovu vrednost */
```



```

        max=m[i][j];
53     /* i cuvamo indeks vrste u kojoj se nalazi */
        indeks_vrste=i;
55     }
    return indeks_vrste;
57 }

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

63     int broj_negativnih=0;

65     for(i=0; i<n; i++)
        for(j=0; j<n; j++)
67         if(m[i][j]<0)
            broj_negativnih++;
69     return broj_negativnih;
}

71
int main(int argc, char* argv[])
73 {
    int m[MAX][MAX];
75     int n;
    int i, j;

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

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

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

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

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

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

```

```
109  /* Prekidamo izvršavanje programa */
      return 0;
111 }
```

### Rešenje 2.12

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

4  #define MAX 32

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

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

16

18  /* Funkcija ispisuje elemente kvadratne matrice na
   standardni izlaz */
20  void ispiši_matricu(int m[][MAX], int n)
   {
22     int i, j;

24     for(i=0; i<n; i++) {
         for(j=0; j<n; j++)
26             printf("%d ", m[i][j]);
         printf("\n");
28     }

30  /* Funkcija proverava da li je zadata matrica ortonormirana */
32  int ortonormirana(int m[][MAX], int n)
   {
34     int i, j, k;
     int proizvod;

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

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

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

46         /* Ako proizvod bar jedne vrste nije jednak jedinici, odmah
           zaključujemo da matrica nije normirana */
48         if(proizvod!=1)
```

```

    return 0;
50 }

52 /* Proveravamo uslov ortogonalnosti, odnosno da li je proizvod
    dve bilo koje razlicite vrste matrice jednak nuli */
54 for(i=0; i<n-1; i++) {
    for(j=i+1; j<n; j++) {
56
58         /* Izracunavamo skalarni proizvod */
        proizvod = 0;

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

62         /* Ako proizvod dve bilo koje razlicite vrste nije jednak
64            nuli, odmah zakljucujemo da matrica nije ortogonalna */
            if(proizvod!=0)
66                 return 0;
        }
68     }

70 /* Ako su oba uslova ispunjena, vracamo jedinicu kao
    rezultat */
72 return 1;
}

74
76 int main()
77 {
    int A[MAX][MAX];
78     int n;

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

84     if( n > MAX || n <= 0) {
        fprintf(stderr, "Greska: neodgovarajuca dimenzija ");
86         fprintf(stderr, "matrice.\n");
        exit(EXIT_FAILURE);
88     }

90     /* Ucitavamo matricu */
    ucitaj_matricu(A, n);

92
94     /* Ispisujemo rezultat rada funkcije */
    if(ortonormirana(A,n))
        printf("da\n");
96     else
        printf("ne\n");
98
100     return 0;
}

```

## Rešenje 2.13

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX_V 10
5  #define MAX_K 10
6
7  /* Funkcija proverava da li su ispisani svi elementi iz matrice,
8     odnosno da li se nariusio prirodan poredak medju granicama */
9  int krajIspisa(int top, int bottom, int left, int right)
10 {
11     return !(top <= bottom && left <= right);
12 }
13
14 /* Funkcija spiralno ispisuje elemente matrice */
15 void ispisi_matricu_spiralno(int a[][MAX_K], int n, int m)
16 {
17     int i,j,top, bottom,left, right;
18
19     top=left = 0;
20     bottom=n-1;
21     right = m-1;
22
23     while( !krajIspisa(top, bottom, left, right) ) {
24         /* Ispisuje se prvi red*/
25         for(j=left; j<=right; j++)
26             printf("%d ",a[top][j]);
27
28         /* Spustamo prvi red */
29         top++;
30
31         if(krajIspisa(top,bottom,left,right))
32             break;
33
34         for(i=top; i<=bottom; i++ )
35             printf("%d ",a[i][right]);
36
37         /* Pomeramo desnu kolonu za naredni krug ispisa
38            blize levom kraju */
39         right--;
40
41         if(krajIspisa(top,bottom,left,right))
42             break;
43
44         /* Ispisujemo donju vrstu */
45         for(j=right; j>=left; j-- )
46             printf("%d ",a[bottom][j]);
47
48         /* Podizemo donju vrstu za naredni krug ispisa */
49         bottom--;
50
51         if(krajIspisa(top,bottom,left,right))
52             break;
53
54         /* Ispisujemo prvu kolonu*/
55         for(i=bottom; i>=top; i-- )
```

```

56     printf("%d ",a[i][left]);
58     /* Pripremamo levu kolonu za naredni krug ispisa */
59     left++;
60 }
61 putchar('\n');
62 }
63
64 void ucitaj_matricu(int a[][MAX_K], int n, int m)
65 {
66     int i, j;
67
68     for(i=0 ;i<n; i++)
69         for(j=0; j<m; j++)
70             scanf("%d", &a[i][j]);
71 }
72
73 int main( )
74 {
75     int a[MAX_V][MAX_K];
76     int m,n;
77
78     /* Ucitaj broj vrsta i broj kolona matrice */
79     scanf("%d",&n);
80     scanf("%d", &m);
81
82     if( n > MAX_V || n <= 0 || m > MAX_K || m <= 0) {
83         fprintf(stderr, "Greska: neodgovarajuće dimenzije ");
84         fprintf(stderr, "matrice.\n");
85         exit(EXIT_FAILURE);
86     }
87
88     ucitaj_matricu(a, n, m);
89     ispisi_matricu_spiralno(a, n, m);
90
91     return 0;
92 }

```

### Rešenje 2.14

### Rešenje 2.15

```

1 #include <stdio.h>
2 #include <stdlib.h>
3
4 /* NAPOMENA: Primer demonstrira dinamičku alokaciju niza od n
5    elemenata. Dovoljno je alocirati n * sizeof(T) bajtova, gde
6    je T tip elemenata niza. Povratnu adresu malloc()-a treba
7    pretvoriti iz void * u T *, kako bismo dobili pokazivac
8    koji pokazuje na prvi element niza tipa T. Na dalje se
9    elementima može pristupati na isti način kao da nam
10   je dato ime niza (koje se tako i ponasa - kao pokazivac
11   na element tipa T koji je prvi u nizu) */
12 int main()

```

```
{
14  int *p = NULL;
16  int i, n;

18  /* 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. */
20  scanf("%d", &n);

22

24  /* Alociramo prostor za n celih brojeva */
    if ((p = (int *) malloc(sizeof(int) * n)) == NULL) {
        fprintf(stderr, "malloc(): ");
26        fprintf(stderr, "greska pri alokaciji memorije.\n");
        exit(EXIT_FAILURE);
28    }

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

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

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

42

44    /* Oslobadjamo prostor */
    free(p);

46    return 0;
}
```

### Rešenje 2.16

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

4
int main(void)
6 {
    /* Adresa prvog alociranog bajta*/
8    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;
19  int* b = NULL;
20
21  /* Inicijalizacija */
22  alocirano = n = 0;
23
24  /* Unosimo brojeve sa ulaza */
25  scanf("%d", &x);
26
27  /* Sve dok je procitani broj razlicit od nule... */
28  while(x!=0) {
29
30      /* Ako broj ucitanih elemenata niza odgovara broju
31         alociranih mesta, za smestanje novog elementa treba
32         obezbediti dodatni prostor. Da se ne bi za svaki sledeci
33         element pojedinačno alocirala memorija, prilikom
34         alokacije se vrši rezervacija za još KORAK dodatnih
35         mesta za buduće elemente */
36      if(n == alocirano) {
37          /* Povecava se broj alociranih mesta */
38          alocirano = alocirano + KORAK;
39
40          /* Vrši se realokacija memorije sa novom velicinom */
41          /******
42          /* Resenje sa funkcijom malloc() */
43          /******
44          /* Vrši se alokacija memorije sa novom velicinom, a adresa
45             početka novog memorijskog bloka se čuva u
46             promenljivoj b */
47          b = (int*) malloc (alocirano * sizeof(int));
48
49          /* Ako prilikom alokacije dodje do neke greske */
50          if(b == NULL) {
51              /* poruku ispisujemo na izlaz za greske */
52              fprintf(stderr, "malloc(): ");
53              fprintf(stderr, "greska pri alokaciji memorije.\n");
54
55              /* Pre kraja programa moramo svu dinamički alociranu
56                 memoriju da oslobodimo. U ovom slučaju samo memoriju
57                 na adresi a */
58              free(a);
59
60              /* Završavamo program */
61              exit(EXIT_FAILURE);
62          }
63
64          /* Svih n elemenata koji počinju na adresi a prepisujemo
65             na novu adresu b */
66          for(i = 0; i < n; i++)
67              b[i] = a[i];
68
69          /* Posle prepisivanja oslobadjamo blok memorije sa početnom
70             adresom u a */
71          free(a);
72
73          /* Promenljivoj a dodeljujemo adresu početka novog, većeg
```

```
74         bloka koji je prilikom alokacije zapamcen u
           promenljivoj b */
76     a = b;

78     /******
       /* Resenje sa funkcijom realloc() */
80     /******
       /* Zbog funkcije realloc je neophodno da i u prvoj
82         iteraciji "a" bude inicijalizovano na NULL */
       /*
84     a = (int*) realloc(a,alocirano*sizeof(int));

86     if(a == NULL) {
           fprintf(stderr, "realloc(): ");
88         fprintf(stderr, "greska pri alokaciji memorije.\n");
           exit(EXIT_FAILURE);
90     }
       */
92 }

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

96
98 /* i učitavamo sledeći element */
scanf("%d", &x);
100 }

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

106 /* Oslobadjamo dinamički alociranu memoriju */
free(a);

108
110 /* Program se završava */
exit(EXIT_SUCCESS);
}
```

### Rešenje 2.17

```
#include <stdio.h>
2 #include <stdlib.h>
#include <string.h>
4
#define MAX 1000
6
/* NAPOMENA: Primer demonstrira "vracanje nizova iz funkcije".
8   Ovako nesto se moze improvizovati tako sto se u funkciji
   dinamički kreira niz potrebne velicine, popuni se potrebnim
10   informacijama, a zatim se vrati njegova adresa. Imajuci u
   vidu cinjenicu da dinamički kreiran objekat ne nestaje
12   kada se izadje iz funkcije koja ga je kreirala, vraceni
   pokazivac se kasnije u pozivajucoj funkciji moze koristiti
14   za pristup "vracenom" nizu. Medjutim, pozivajuca funkcija
```



```
16     ima odgovornost i da se brine o dealokaciji istog prostora */
18 /* Funkcija dinamički kreira niz karaktera u koji smesta
20 rezultat nadovezivanja niski. Adresa niza se vraća kao
    povratna vrednost. */
22 char *nadovezi(char *s, char *t) {
    /* Dinamički kreiramo prostor dovoljne velicine */
    char *p = (char *) malloc((strlen(s) + strlen(t) + 1)
                               * sizeof(char));
24
    /* Proveravamo uspeh alokacije */
26 if (p == NULL) {
    fprintf(stderr, "malloc(): ");
28 fprintf(stderr, "greska pri alokaciji memorije.\n");
    exit(EXIT_FAILURE);
30 }

32 /* Kopiramo i nadovezujemo stringove */

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

40 for(j=0; t[j]!='\0'; i++, j++)
    p[i]=t[j];
42
    p[i]='\0';
44 */

46 /* Resenje sa koriscenjem biblioteckih funkcija iz zaglavlja
    string.h */
48 strcpy(p, s);
    strcat(p, t);
50
    /* Vracamo pokazivac p */
52 return p;
    }
54

56 int main() {
    char *s = NULL;
    char s1[MAX], s2[MAX];
58
    /* Ucitavamo dve niske koje cemo da nadovezemo */
60 scanf("%s", s1);
    scanf("%s", s2);
62
    /* Pozivamo funkciju da nadoveze stringove */
64 s = nadovezi(s1, s2);

66 /* Prikazujemo rezultat */
    printf("%s\n", s);
68
    /* Oslobadjamo memoriju alociranu u funkciji nadovezi() */
70 free(s);
```

```
72     return 0;
    }
```

### Rešenje 2.18

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

int main()
{
    int i,j;

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

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

    /* Trag matrice */
    double trag = 0;

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

    /* Dinamički 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");
        exit(EXIT_FAILURE);
    }

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

        if(A[i] == NULL) {
            /* Alokacija je neuspesna. Pre zavrsetka programa
               moramo da oslobodimo svih i-1 prethodno alociranih
               vrsta, i alociran niz pokazivaca */
            for( j=0; j<i; j++)
                free(A[j]);
            free(A);

            exit( EXIT_FAILURE);
        }
    }

    /* Unosimo sa standardnog ulaza brojeve u matricu.
       Popunjavamo vrstu po vrstu */
```

```

50     for(i = 0; i < n; i++)
51         for(j = 0; j < m; j++ )
52             scanf("%lf", &A[i][j]);

54     /* Racunamo trag matrice, odnosno sumu elemenata
       na glavnoj dijagonali */
56     trag = 0.0;

58     for(i=0; i<n; i++)
59         trag += A[i][i];

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

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

66     /* Oslobadjamo memoriju za niz pokazivaca na vrste */
68     free(A);

70     return 0;
}

```

### Rešenje 2.19

```

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

5  void ucitaj_matricu(int ** M, int n, int m)
6  {
7      int i, j;

9      /* Popunjavamo matricu vrstu po vrstu */
10     for(i=0; i<n; i++)
11         /* Popunjavamo i-tu vrstu matrice */
12         for(j=0; j<m; j++)
13             scanf("%d", &M[i][j]);
14 }

15 void ispisuje_elemente_ispod_dijagonale(int ** M, int n, int m)
16 {
17     int i, j;

19     for(i=0; i<n; i++) {
20         /* Ispisujemo elemente ispod glavne dijagonale matrice */
21         for(j=0; j<=i; j++)
22             printf("%d ", M[i][j]);
23         printf("\n");
24     }
25 }

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

```

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

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

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

45     if(matrica[i] == NULL) {
47       fprintf(stderr,"malloc(): Neuspela alokacija\n");
       for(j=0; j<i; j++)
49         free(matrica[j]);
       free(matrica);
51       exit(EXIT_FAILURE);
     }
53   }

55   ucitaj_matricu(matrica, n, m);

57   ispisi_elemente_ispod_dijagonale(matrica, n, m);

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

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

67   return 0;
69 }
```

### Rešenje 2.20

```
#include<stdio.h>

2
int main(){
4   printf("Hello pokazivaci!\n");
   return 0;
6 }
```

### Rešenje 2.21

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

```

4
/* Funkcija izvrsava trazene transformacije nad matricom */
6 void izmeni (float** a, int n)
{
8     int i, j;

10    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
12    /* Ako je indeks vrste manji od indeksa kolone */
        if(i<j)
14        /* element se nalazi iznad glavne dijagonale
            pa ga polovimo */
            a[i][j]/=2;
16        else
18        /* Ako je indeks vrste veci od indeksa kolone */
            if(i>j)
20            /* element se nalazi ispod glavne dijagonale
                pa ga dupliramo*/
                a[i][j] *= 2;
22    }

24    /* Funkcija izracunava zbir apsolutnih vrednosti elemenata ispod
26    sporedne dijagonale */
    float zbir_ispod_sporedne_dijagonale(float** m, int n)
28    {
        int i, j;
30        float zbir=0;

32        for(i=0; i<n; i++)
            for(j=0; j<n; j++)
34            /* Ukoliko je zbir indeksa vrste i indeksa kolone
                elementa veci od n-1, to znaci da se element nalazi
36            ispod sporedne dijagonale */
                if(i+j>n-1)
38                zbir+=fabs(m[i][j]);

40        return zbir;
    }

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

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

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

58        for(i=0; i<n; i++){
            for(j=0; j<n; j++)

```

```
60     printf("%.2f ", m[i][j]);
61     printf("\n");
62 }
63 }
64
65 /* Funkcija alokira memoriju za kvadratnu matricu dimenzije n */
66 float** alociraj_memoriju(int n) {
67     int i, j;
68     float** m;
69
70     m = (float**) malloc(n * sizeof(float*));
71     if(m == NULL) {
72         fprintf(stderr, "malloc(): Neuspela alokacija\n");
73         exit(EXIT_FAILURE);
74     }
75
76     /* Za svaku vrstu matrice */
77     for(i=0; i<n; i++) {
78         /* Alociramo memoriju */
79         m[i] = (float*) malloc(n * sizeof(float));
80
81         /* Proveravamo da li je doslo do greske pri alokaciji */
82         if(m[i] == NULL) {
83             /* Ako jeste, ispisujemo poruku */
84             printf("malloc(): neuspela alokacija memorije!\n");
85
86             /* Oslobadjamo memoriju zauzetu do ovog koraka */
87             for(j=0; j<i; j++)
88                 free(m[j]);
89             free(m);
90             exit(EXIT_FAILURE);
91         }
92     }
93     return m;
94 }
95
96 /* Funkcija oslobadja memoriju zauzetu kvadratnom matricom
97    dimenzije n */
98 void oslobodi_memoriju(float** m, int n)
99 {
100     int i;
101
102     for(i=0; i<n; i++)
103         free(m[i]);
104     free(m);
105 }
106
107 int main(int argc, char* argv[])
108 {
109     FILE* ulaz;
110     float** a;
111     int n;
112
113     /* Ako korisnik nije uneo trazene argumente,
114        prijavljujemo gresku */
115     if(argc < 2) {
```

```
116     printf("Greska: ");
117     printf("Nedovoljan broj argumenata komandne linije.\n");
118     printf("Program se poziva sa %s ime_dat.\n", argv[0]);
119     exit(EXIT_FAILURE);
120 }
121
122 /* Otvaramo datoteku za citanje */
123 ulaz = fopen(argv[1], "r");
124 if(ulaz == NULL) {
125     fprintf(stderr, "Greska: ");
126     fprintf(stderr, "Neuspesno otvaranje datoteke %s.\n",
127                                     argv[1]);
128     exit(EXIT_FAILURE);
129 }
130
131 /* citamo dimenziju matrice */
132 fscanf(ulaz, "%d", &n);
133
134 /* Alociramo memoriju */
135 a = alociraj_memoriju(n);
136
137 /* Ucitavamo elemente matrice */
138 ucitaj_matricu(ulaz, a, n);
139
140 float zbir = zbir_ispod_sporedne_dijagonale(a, n);
141
142 /* Pozivamo funkciju za modifikovanje elemenata */
143 izmeni(a, n);
144
145 /* Ispisujemo rezultat */
146 printf("Zbir apsolutnih vrednosti ispod sporedne dijagonale ");
147 printf("je %.2f.\n", zbir);
148
149 printf("Transformisana matrica je:\n");
150 ispisi_matricu(a,n);
151
152 /* Oslobadjamo memoriju */
153 oslobodi_memoriju(a, n);
154
155 /* Zatvaramo datoteku */
156 fclose(ulaz);
157
158 /* i prekidamo sa izvršavanjem programa */
159 return 0;
160 }
```

Rešenje [2.22](#)

Rešenje [2.23](#)

Rešenje [2.24](#)

Rešenje [2.25](#)

## Rešenje 2.26

```
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <math.h>
5 #include <string.h>
6
7 /* NAPOMENA:
8  Zaglavlje math.h sadrži deklaracije raznih matematičkih
9  funkcija. Izmeđ ostalog, to su sledeće funkcije:
10 double sin(double x);
11 double cos(double x);
12 double tan(double x);
13 double asin(double x);
14 double acos(double x);
15 double atan(double x);
16 double atan2(double y, double x);
17 double sinh(double x);
18 double cosh(double x);
19 double tanh(double x);
20 double exp(double x);
21 double log(double x);
22 double log10(double x);
23 double pow(double x, double y);
24 double sqrt(double x);
25 double ceil(double x);
26 double floor(double x);
27 double fabs(double x);
28 */
29
30 /* Funkcija tabela() prihvata granice intervala a i b, broj
31 ekvidistantnih čtaaka n, kao i pokaziva f koji pokazuje
32 na funkciju koja prihvata double argument, i čvraa double
33 vrednost. Za tako datu funkciju ispisuje njene vrednosti
34 u intervalu [a,b] u n ekvidistantnih čtaaka intervala */
35 void tabela(double a, double b, int n, double (*fp)(double))
36 {
37     int i;
38     double x;
39
40     printf("-----\n");
41     for(i=0; i<n; i++) {
42         x= a + i*(b-a)/(n-1);
43         printf("| %8.5f | %8.5f |\n", x, (*fp)(x));
44     }
45     printf("-----\n");
46 }
47
48 /* Umesto da koristimo stepenu funkciju iz zaglavlja
49 math.h -> pow(a,2) čpozivaemo čkorisniku sqr(a) */
50 double sqr (double a)
51 {
52     return a*a;
53 }
54
```



```

int main(int argc, char *argv[])
{
    double a, b;
    int n;
    /* Imena funkcija koja ćemo navoditi su čkraa 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)
        fp=&tan;
    else if(strcmp(ime_fje, "atan") == 0)
        fp=&atan;
    else if(strcmp(ime_fje, "acos") == 0)
        fp=&acos;
    else if(strcmp(ime_fje, "asin") == 0)
        fp=&asin;
    else if(strcmp(ime_fje, "exp") == 0)
        fp=&exp;
    else if(strcmp(ime_fje, "log") == 0)
        fp=&log;
    else if(strcmp(ime_fje, "log10") == 0)
        fp=&log10;
    else if(strcmp(ime_fje, "sqrt") == 0)
        fp=&sqrt;
    else if(strcmp(ime_fje, "floor") == 0)
        fp=&floor;
    else if(strcmp(ime_fje, "ceil") == 0)
        fp=&ceil;
    else if(strcmp(ime_fje, "sqr") == 0)
        fp=&sqr;
    else {
        printf("Program jos uvek ne podrzava trazenu funkciju!\n");
        exit(EXIT_SUCCESS);
    }
}

```

```
112     }
113
114     printf("Unesite krajeve intervala:\n" );
115     scanf("%lf %lf", &a, &b);
116
117     printf("Koliko tacaka ima na ekvidistantnoj mrezi ");
118     printf("(ukljucujuci krajeve intervala)?\n");
119     scanf("%d", &n );
120
121     /* Mreza mora da čukljuuje bar krajeve intervala,
122        tako da se mora uneti broj veci od 2 */
123     if (n < 2) {
124         fprintf(stderr, "Broj čtaaka žmree mora biti bar 2!\n");
125         exit(EXIT_FAILURE);
126     }
127
128     /* Ispisujemo ime funkcije */
129     printf("      x %10s(x)\n", ime_fje);
130
131     /* dProsleujemo funkciji tabela() funkciju zadatu kao
132        argument komandne linije */
133     tabela(a, b, n, fp);
134
135     exit(EXIT_SUCCESS);
136 }
```

Rešenje [2.27](#)

Rešenje [2.28](#)

Rešenje [2.29](#)

## Glava 3

# Algoritmi pretrage i sortiranja

### 3.1 Pretraživanje

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

- (a) Napisati funkciju koja vrši linearnu pretragu niza celih brojeva  $a$ , dužine  $n$ , tražeći u njemu broj  $x$ .
- (b) Napisati funkciju koja vrši binarnu pretragu sortiranog niza  $a$ , dužine  $n$ , tražeći u njemu broj  $x$ .
- (c) Napisati funkciju koja vrši interpoacionu pretragu sortiranog niza  $a$ , dužine  $n$ , tražeći u njemu broj  $x$ .

Napisati i program koji generiše slučajni rastući niz dimenzije  $n$  (prvi argument komandne linije), i u njemu već napisanim funkcijama traži element  $x$  (drugi argument komandne linije). Potrebna vremena za izvršavanje ovih funkcija upisati u fajl `vremena.txt`.

*Test 1*

```
Poziv: ./a.out 1000000 235423
```

```
Izlaz:
```

```
Linearna pretraga
```

```
Element nije u nizu
```

```
-----
```

```
Binarna pretraga
```

```
Element nije u nizu
```

```
-----
```

```
Interpolaciona pretraga
```

```
Element nije u nizu
```

```
-----
```

**Zadatak 3.2** Napisati rekurzivne funkcije algoritama linearne, binarne i interpolacione pretrage i program koji ih testira za brojeve koji se unose sa standardnog

### 3 Algoritmi pretrage i sortiranja

---

ulaza. Pretpostaviti da niz brojeva koji se unosi neće biti duži od 1024 elemenata. Prvo se unosi broj koji se traži, a zatim sortirani elementi niza sve do kraja ulaza.

<i>Test 1</i>	<i>Test 2</i>
<code>Ulaz: 11 2 5 6 8 10 11 23</code>	<code>Ulaz: 14 10 32 35 43 66 89 100</code>
<code>Izlaz:</code>	<code>Izlaz:</code>
<code>Linearna pretraga</code>	<code>Linearna pretraga</code>
<code>Pozicija elementa je 5.</code>	<code>Element se ne nalazi u nizu.</code>
<code>Binarna pretraga</code>	<code>Binarna pretraga</code>
<code>Pozicija elementa je 5.</code>	<code>Element se ne nalazi u nizu.</code>
<code>Interpolaciona pretraga</code>	<code>Interpolaciona pretraga</code>
<code>Pozicija elementa je 5.</code>	<code>Element se ne nalazi u nizu.</code>

**Zadatak 3.3** Napisati program koji preko argumenta komandne linije dobija ime datoteke koja sadrži sortirani spisak studenta po broju indeksa rastuće. Za svakog studenta u jednom redu stoje informacije o indeksu, imenu i prezimenu. Program učitava spisak studenata u niz i traži od korisnika indeks studenta čije informacije se potom prikazuju na ekranu. Zatim, korisnik učitava prezime studenta i prikazuju mu se informacije o prvom studentu sa unetim prezimenom. Pretrage implementirati u vidu iterativnih funkcija što bolje manje složenosti. Pretpostaviti da u datoteci neće biti više od 128 studenata, i da su imena i prezimena svih kraća od 16 slova.

<i>Test 1</i>
<code>Datoteka:</code>
<code>20140003 Marina Petrovic</code>
<code>20140012 Stefan Mitrovic</code>
<code>20140032 Dejan Popovic</code>
<code>20140049 Mirko Brankovic</code>
<code>20140076 Sonja Stevanovic</code>
<code>20140104 Ivan Popovic</code>
<code>20140187 Vlada Stankovic</code>
<code>20140234 Darko Brankovic</code>
<code>Ulaz:</code>
<code>20140076</code>
<code>Popovic</code>
<code>Izlaz:</code>
<code>Indeks: 20140076, Ime i prezime: Sonja Stevanovic</code>
<code>Indeks: 20140032, Ime i prezime: Dejan Popovic</code>

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

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

*Test 1*

```

|| Poziv: ./a.out dat.txt -x
|| Datoteka:
|| 12 53
|| 2.342 34.1
|| -0.3 23
|| -1 23.1
|| 123.5 756.12
|| Izlaz: -0.3 23

```

**Zadatak 3.6** Napisati funkciju koja određuje nulu funkcije  $\cos(x)$  na intervalu  $[0, 2]$  metodom polovljenja intervala. Algoritam se završava kada se vrednost kosinusne funkcije razlikuje za najviše 0.001 od nule. Uputstvo: koristiti algoritam analogan algoritmu binarne pretrage.

*Test 1*

```

|| Izlaz: 1.57031

```

**Zadatak 3.7** Napisati funkciju koja u rastuće sortiranom nizu celih brojeva binarnom pretragom pronalazi prvi element veći od nule i kao rezultat vraća njegovu poziciju u nizu. Ukoliko nema elemenata većih od nule, funkcija kao rezultat vraća -1. Napisati program koji testira ovu funkciju za niz elemenata koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

*Test 1*

```

|| Ulaz: -151 -44 5
|| 12 13 15
|| Izlaz: 2

```

*Test 2*

```

|| Ulaz: -100 -15 -11
|| -8 -7 -5
|| Izlaz: -1

```

*Test 3*

```

|| Ulaz: -100 -15 0 13
|| 155 124 258
|| 315 516 7000
|| Izlaz: 3

```

**Zadatak 3.8** Napisati funkciju koja u opadajuće sortiranom nizu celih brojeva binarnom pretragom pronalazi prvi element manji od nule i kao rezultat vraća njegovu poziciju u nizu. Ukoliko nema elemenata manjih od nule, funkcija kao rezultat vraća -1. Napisati program koji testira ovu funkciju za niz elemenata koji se učitavaju sa standardnog ulaza. Niz neće biti duži od 256, i njegovi elementi se unose sve do kraja ulaza.

*Test 1*

```

|| Ulaz: 151 44 5 -12 -13 -15
|| Izlaz: 3

```

*Test 2*

```

|| Ulaz: 100 55 15 0 -15 -124 -155
|| -258 -315 -516 -7000
|| Izlaz: 4

```

*Test 3*

```
|| Ulaz:  100 15 11 8 7 5 4 3 2
|| Izlaz: -1
```

**Zadatak 3.9** Napisati funkciju koja određuje ceo deo logaritma za osnovu 2 datog neoznačenog celog broja, koristeći samo bitske i relacione operatore.

- (a) Napisati funkciju, linearne složenosti, koja određuje logaritam pomeranjem broja udesno dok ne postane 0.
- (b) Napisati funkciju, logaritmske složenosti, koja određuje logaritam koristeći binarnu pretragu.

Tražene funkcije testirati programom koji broj učitava sa standardnog ulaza, a logaritam ispisuje na standardni izlaz.

*Test 1*

```
|| Ulaz:  10
|| Izlaz: 3 3
```

*Test 2*

```
|| Ulaz:  4
|| Izlaz: 2 2
```

*Test 3*

```
|| Ulaz:  17
|| Izlaz: 4 3
```

*Test 4*

```
|| Ulaz:  1031
|| Izlaz: 10 10
```

**\*\* Zadatak 3.10** U prvom kvadrantu dato je  $1 \leq N \leq 10000$  duži svojim koordinatama (duži mogu da se seku, preklapaju, itd.). Napisati program koji pronalazi najmanji ugao  $0 \leq \alpha \leq 90^\circ$ , na dve decimale, takav da je suma dužina duži sa obe strane polupoluprave iz koordinatnog početka pod uglom  $\alpha$  jednak (neke duži bivaju presečene, a neke ne). Program prvo učitava broj  $N$ , a zatim i same koordinate temena duži. Uputstvo: vršiti binarnu pretragu intervala  $[0, 90^\circ]$ .

*Test 1*

```
|| Ulaz:
|| 2
|| 2 0 2 1
|| 1 2 2 2
|| Izlaz: 26.57
```

**Zadatak 3.11** Napisati program u kome se prvo inicijalizuje statički niz struktura osoba sa članovima ime i prezime (uređen u rastućem poretку prezimena) sa manje od 10 elemenata, a zatim se učitava jedan karakter i pronalazi (bibliotečkom funkcijom `bsearch`) i štampa jedna struktura iz niza osoba čije prezime počinje tim karakterom. Ako takva osoba ne postoji, štampati  $-1$  na standardni izlaz.

```
Osoba niz_osoba[]={{"Mika", "Antic"},
                   {"Dobrica", "Eric"},
```

```

{"Desanka", "Maksimovic"},
{"Dusko", "Radovic"},
{"Ljubivoje", "Rsumovic"};

```

*Test 1*

```

|| Ulaz:  R
|| Izlaz: Dusko Radovic

```

## 3.2 Sortiranje

**Zadatak 3.12** U datom nizu brojeva pronaći dva broja koja su na najmanjem rastojanju. Niz se zadaje sa standardnog ulaza, sve do kraja ulaza, i neće sadržati više od 256 elemenata. Na izlaz ispisati njihovu razliku. Uputstvo: prvo sortirati niz.

*Test 1*

```

|| Ulaz:  23 64 123 76 22 7
|| Izlaz:  1

```

*Test 2*

```

|| Ulaz:  21 654 65 123 65 12 61
|| Izlaz:  0

```

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

*Test 1*

```

|| Ulaz:  anagram ramgana
|| Izlaz:  jesu

```

*Test 2*

```

|| Ulaz:  anagram anagrm
|| Izlaz:  nisu

```

**Zadatak 3.14** Napisati program koji pronalazi broj koji se najviše puta pojavljivao u datom nizu. Niz se zadaje sa standardnog ulaza sve do kraja ulaza, i neće biti duži od 256 elemenata. Uputstvo: prvo sortirati niz, a zatim naći najdužu sekvencu jednakih elemenata.

*Test 1*

```

|| Ulaz:  4 23 5 2 4 6 7 34 6 4 5
|| Izlaz:  4

```

*Test 2*

```

|| Ulaz:  2 4 6 2 6 7 99 1
|| Izlaz:  2

```

### 3 Algoritmi pretrage i sortiranja

---

**Zadatak 3.15** Napisati funkciju koja proverava da li u datom nizu postoje dva elementa kojima je zbir zadati ceo broj. Napisati i program koji testira ovu funkciju. U programu se prvo učitava broj, a zatim i niz (pretpostaviti da za niz neće biti uneto više od 256 brojeva). Elementi niza se unose sve do kraja ulaza. Uputstvo: prvo sortirati niz.

	<i>Test 1</i>		<i>Test 2</i>
	Ulaz: 34 134 4 1 6 30 23		Ulaz: 12 53 1 43 3 56 13
	Izlaz: da		Izlaz: ne

**Zadatak 3.16** Napisati funkciju potpisa `int merge(int *niz1, int dim1, int *niz2, int dim2, int *niz3, int dim3)` koja prima dva sortirana niza, i na osnovu njih pravi novi sortirani niz koji koji sadrži elemente oba niza. Treća dimenzija predstavlja veličinu niza u koji se smešta rezultat. Ako je ona manja od potrebne dužine, funkcija vraća -1, kao indikator neuspeha, inače vraća 0. Napisati i program koji testira funkciju, u kome se nizovi unose sa standardnog ulaza, sve dok se ne unese 0. Dimenzija nizova neće biti preko 256.

	<i>Test 1</i>
	Ulaz: 3 6 7 11 14 35 0 3 5 8 0
	Izlaz: 3 3 5 6 7 8 11 14 35

	<i>Test 2</i>
	Ulaz: 1 4 7 0 9 11 23 54 75 0
	Izlaz: 1 4 7 9 11 23 54 75

**Zadatak 3.17** Napisati program koji čita sadržaj dve datoteke od kojih svaka sadrži spisak imena i prezimena studenata iz jedne od dve grupe, rastuće sortiran po imenima i kreira jedinstven spisak studenata sortiranih takođe po imenu rastuće. Program dobija nazive datoteka iz komandne linije, i jedinstven spisak upisuje u datoteku `ceo-tok.txt`. Pretpostaviti da je ime studenta nije duže od 10, a prezime od 15 karaktera.

	<i>Test 1</i>
	Poziv: ./a.out prvi-deo.txt drugi-deo.txt
	prvi-deo.txt:                      drugi-deo.txt:                      ceo-tok.txt:
	Andrija Petrovic                      Aleksandra Cvetic                      (TODO)
	Anja Ilic                              Bojan Golubovic
	Ivana Markovic                      Dragan Markovic
	Lazar Micic                              Filip Dukic
	Nenad Brankovic                      Ivana Stankovic
	Sofija Filipovic                      Marija Stankovic
	Vladimir Savic                      Ognjen Peric
	Uros Milic



**Zadatak 3.18** Napraviti biblioteku `sort.h` i `sort.c` koja implementira algoritme sortiranja nizova celih brojeva. Biblioteka treba da sadrži `selection`, `merge`, `quick`, `bubble`, `insertion` i `shell sort`. Upotrebiti biblioteku kako bi se napravilo poređenje efikasnosti različitih algoritama sortiranja. Efikasnost meriti na slučajno generisanim nizovima, na već sortiranim nizovima i na naopako sortiranim nizovima. Izbor algoritma, veličine i početnog rasporeda elemenata niza birati kroz argumente komandne linije. Vreme meriti programom `time`. Analizirati porast vremena sa porastom dimenzije `n`.

**Zadatak 3.19** Napisati funkcije koje sortiraju niz struktura tačaka na osnovu sledećih kriterijuma:

- (a) njihovog rastojanja od koordinatnog početka,
- (b) `x` koordinata tačaka,
- (c) `y` koordinata tačaka.

Napisati program koji učitava niz tačaka iz datoteke čije se ime zadaje kao argument komandne linije, i u zavisnosti od prisutnih opcija u komandnoj liniji (`-d`, `-x` ili `-y`), sortira tačke po jednom od prethodna tri kriterijuma i rezultat upisuje u datoteku čije se ime zadaje kao drugi argument komandne linije. U ulaznoj datoteci nije zadato više od 128 tačaka.

#### Test 1

```
Poziv:  a.out -x tacke.txt
        sorttacke.txt
Ulazna datoteka:
3 4
11 6
7 3
2 82
-1 6
Izlazna datoteka:
-1 6
2 82
3 4
7 3
11 6
```

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

#### Test 1

```
biracki-spisak.txt:      Izlaz:
Andrija Petrovic        (TODO)
Anja Ilic
Aleksandra Cvetic
Bojan Golubovic
Dragan Markovic
Filip Dukic
Ivana Stankovic
Ivana Markovic
Lazar Micic
Marija Stankovic
```

**Zadatak 3.21** Definisana je struktura podataka

```
typedef struct dete
{
    char ime[MAX_IME];
    char prezime[MAX_IME];
    unsigned godiste;
} Dete;
```

Napisati funkciju koja sortira niz dece po godištu, a kada su deca istog godišta, tada ih sortira leksikografski po prezimenu i imenu. Napisati program koji učitava podatke o deci koji se nalaze u datoteci, čije se ime zadaje kao prvi argument komandne linije, sortira ih i sortirani niz upisuje u datoteku čije se ime zadaje kao drugi argument komandne linije. Pretpostaviti da u ulaznoj datoteci nisu zadati podaci o više od 128 dece.

#### Test 1

```
Poziv: ./a.out ulaz.txt izlaz.txt
Ulazna datoteka:      Izlazna datoteka:
Petar Petrovic 2007    Marija Antic 2007
Milica Antonic 2008    Ana Petrovic 2007
Ana Petrovic 2007      Petar Petrovic 2007
Ivana Ivanovic 2009    Milica Antonic 2008
Dragana Markovic 2010  Ivana Ivanovic 2009
Marija Antic 2007      Dragana Markovic 2010
```

**Zadatak 3.22** Napisati funkciju koja sortira niz niski po broju suglasnika u niski, ukoliko reči imaju isti broj suglasnika tada po dužini niske, a ukoliko su i dužine jednake onda leksikografski. Napisati program koji testira ovu funkciju za niske koje se zadaju u datoteci `niske.txt`. Pretpostaviti da u nizu nema više od 128 elemenata, kao i da svaka niska sadrži najviše 32 karaktera.

*Test 1*

```
Ulazna datoteka:
ana petar andjela milos nikola aleksandar ljubica matej milica
Izlaz:
ana matej milos petar milica nikola andjela ljubica aleksandar
```

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

*Test 1*

<code>artikli.txt:</code>	<code>Ulaz:</code>	<code>Izlaz:</code>
1001 Keks Jaffa 120	(TODO)	(TODO)
2530 Napolitanke Bambi 230		
0023 Medeno_srce Pionir 150		
2145 Pardon Marbo 70		

**Zadatak 3.24** Napisati program koji iz datoteke `aktivnost.txt` čita podatke o aktivnosti studenata na praktikumima i u datoteke `dat1.txt`, `dat2.txt` i `dat3.txt` upisuje redom tri spiska. Na prvom su studenti sortirani leksikografski po imenu rastuće. Na drugom su sortirani po ukupnom broju urađenih zadataka opadajuće, a ukoliko neki studenti imaju isti broj rešenih zadataka sortiraju se po dužini imena rastuće. Na trećem spisku kriterijum sortiranja je broj časova na kojima su bili opadajuće. Ukoliko neki studenti imaju isti broj časova, sortirati ih opadajuće po broju urađenih zadataka, a ukoliko se i on poklapa sortirati po prezimenu opadajuće. U datoteci se nalazi ime, prezime studenta, broj časova na kojima je prisustvovao, kao i ukupan broj urađenih zadataka. Pretpostaviti da studenata neće biti više od 500 i da je za ime studenta dovoljno 20, a za prezime 25 karaktera.

#### Test 1

```
aktivnosti.txt:
  Izlaz:
Aleksandra Cvetic 4 6      (
  TODO)
Bojan Golubovic 4 3
Dragan Markovic 3 5
Filip Dukic 2 0
Ivana Stankovic 3 1
Marija Stankovic 1 3
Ognjen Peric 1 2
Uros Milic 2 5
Andrija Petrovic 2 5
Anja Ilic 3 1
Ivana Markovic 2 5
Lazar Micic 1 3
Nenad Brankovic 2 4
```

**Zadatak 3.25** U datoteci „pesme.txt“ nalaze se informacije o gledanosti pesama na Youtube-u. Format datoteke sa informacijama je sledeći:

- U prvoj liniji datoteke se nalazi ukupan broj pesama prisutnih u datoteci.
- Svaki naredni red datoteke sadrži informacije o gledanosti pesama u formatu izvođač - naslov, broj gledanja.

Napisati program koji učitava informacije o pesmama i vrši sortiranje pesama u zavisnosti od argumenata komandne linije na sledeći način:

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

Na standardni izlaz ispisati informacije o pesmama sortirane na opisan način.

- Uraditi zadatak uz pretpostavku da je maksimalna dužina imena izvođača 20 karaktera, a imena naslova pesme 50 karaktera.
- Uraditi zadatak bez pravljenja pretpostavki o maksimalnoj dužini imena izvođača i naslova pesme.

Jelena: Kako ovde da prikazem rešenje pod a) i b)? Milena: Mislim da bi bilo najbolje razdvojiti kod po datotekama i napisati dve main funkcije. Na taj način onda nedupliramo dva programa, već oba uključuju zajednicke delove, ukoliko ima dovoljno tih zajednickih delova. Ukoliko ih ima malo, onda svaki zadatak posebno.

*Test 1*

```

Poziv: ./a.out
Datoteka: 5
        Ana - Nebo, 2342
        Laza - Oblaci, 29
        Pera - Ptice, 327
        Jelena - Sunce, 92321
        Mika - Kisa, 5341
Izlaz:  Jelena - Sunce, 92321
        Mika - Kisa, 5341
        Ana - Nebo, 2342
        Pera - Ptice, 327
        Laza - Oblaci, 29

```

*Test 2*

```

Poziv: ./a.out -i
Datoteka: 5
        Ana - Nebo, 2342
        Laza - Oblaci, 29
        Pera - Ptice, 327
        Jelena - Sunce, 92321
        Mika - Kisa, 5341
Izlaz:  Ana - Nebo, 2342
        Jelena - Sunce, 92321
        Laza - Oblaci, 29
        Mika - Kisa, 5341
        Pera - Ptice, 327

```

*Test 3*

```

Poziv: ./a.out -n
Datoteka: 5
        Ana - Nebo, 2342
        Laza - Oblaci, 29
        Pera - Ptice, 327
        Jelena - Sunce, 92321
        Mika - Kisa, 5341
Izlaz:  Mika - Kisa, 5341
        Ana - Nebo, 2342
        Laza - Oblaci, 29
        Pera - Ptice, 327
        Jelena - Sunce, 92321

```

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

*Test 1*

```

Ulaz: U 2 U 0 U 6 U 4 N 1 U 8 N 2 N 5 U 2 N 3 N 5
Izlaz: 0 2 8 2 6

```

**\*\* Zadatak 3.27** Šef u restoranu je neuredan i palačinke koje ispeče ne slaže redom po veličini. Konobar pre serviranja mora da sortira palačinke po veličini, a jedina operacija koju sme da izvodi je da obrne deo palačinki. Na primer, sledeća slika po kolonama predstavlja naslagane palačinke posle svakog okretanja. Na početku, palačinka veličine 2 je na dnu, iznad nje se redom nalaze najmanja, najveća, itd... Na slici crtica predstavlja mesto iznad koga će konobar okrenuti palačinke. Prvi potez konobara je okretanje palačinki veličine 5, 4 i 3 (prva kolona), i tada

će veličine palačinki odozdo nagore biti 2, 1, 3, 4, 5 (druga kolona). Posle još dva okretanja, palačinke će biti složene.

3	5	2	1
4	4	1__	2
5__	3	3	3
1__	1	4	4
2	2__	5	5

Napisati program koji u najviše  $2n-3$  okretanja sortira učitani niz. Uputstvo: imitirati `selection sort` i u svakom koraku dovesti jednu palačinku na svoje mesto korišćenjem najviše dva okretanja.

## 3.3 Bibliotečke funkcije pretrage i sortiranja

**Zadatak 3.28** Napisati program koji ilustruje upotrebu bibliotečkih funkcija za pretraživanje i sortiranje nizova, i mogućnost zadavanja različitih kriterijuma sortiranja. Sa standardnog ulaza se unosi dimenzija niza celih brojeva (ne veća od 100), a potom i sami elementi niza. Upotrebom funkcije `qsort` sortirati niz u rastućem poretku, sa standardnog ulaza učitati broj koji se traži u nizu, pa zatim funkcijama `bsearch` i `lfind` utvrditi da li se zadati broj nalazi u nizu i na standardni izlaz ispisati odgovarajuću poruku.

*Test 1*

|| `TODO`

**Zadatak 3.29** Napisati program koji sa standardnog ulaza učitava dimenziju niza celih brojeva (ne veću od 100), a potom i same elemente niza. Upotrebom funkcije `qsort` sortirati niz u rastućem poretku prema broju delilaca i tako dobijeni niz odštampati na standardni izlaz.

*Test 1*

|| `TODO`

**Zadatak 3.30** Korišćenjem bibliotečke funkcije `qsort` napisati program koji sortira niz niski po sledećim kriterijumima:

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

Niske se učitavaju iz fajla `niske.txt`, neće ih biti više od 1000, i svaka će biti dužine najviše 30 karaktera. Program prvo leksikografski sortira niz, primenjuje binarnu pretragu (`bsearch`) zarad traženja niske unete sa standardnog ulaza, a potom linearnu pretragu koristeći funkciju `lfind`. Na kraju, niske bivaju sortirane po dužini. Rezultate svih sortiranja i pretraga ispisati na standardni izlaz.

*Test 1*

|| `TODO`

**Zadatak 3.31** Uraditi prethodni zadatak 3.30 sa dinamički alociranim niskama, i sortiranjem niza pokazivača (umesto niza niski).

*Test 1*

|| `TODO`

**Zadatak 3.32** Napisati program koji korišćenjem bibliotečke funkcije `qsort` sortira studente prema broju poena osvojenih na kolokvijumu. Ukoliko više studenata ima isti broj bodova, sortirati ih po prezimenu leksikografski rastuće. Korisnik potom unosi broj bodova i prikazuje mu se jedan od studenata sa tim brojem bodova, ili poruka ukoliko nema takvog. Potom, sa standardnom ulaza, unosi se prezime traženog studenta, i prikazuje se osoba sa tim prezimenom, ili poruka da se nijedan student tako ne preziva. Za pretraživanje, koristiti odgovarajuće bibliotečke funkcije. Podaci o studentima čitaju se iz datoteke čije se ime zadaje preko argumenata komandne linije. Za svakog studenta u datoteci postoje ime, prezime i bodovi osvojeni na kolokvijumu. Pretpostaviti da neće biti više od 500 studenata, i da je za ime i prezime svakog studenta dovoljno po 20 karaktera.

*Test 1*

|| `TODO`

**Zadatak 3.33** Uraditi zadatak 3.13, ali korišćenjem bibliotečke `qsort` funkcije.

*Test 1*

*Test 2*

<code>Ulaz: vrata vatra</code>	<code>Ulaz: qsort bsearch</code>
<code>Izlaz: jesu</code>	<code>Izlaz: nisu</code>

**Zadatak 3.34** Napisati program koji sa standardnog ulaza učitava prvo ceo broj  $n$  ( $n \leq 10$ ), a zatim niz `S` od  $n$  stringova (maksimalna dužina stringa je 32 karaktera). Sortirati niz `S` (bibliotečkom funkcijom `qsort`) i proveriti da li u njemu ima identičnih stringova.

*Test 1*

*Test 2*

<code>Ulaz: 4 prog search sort search</code>	<code>Ulaz: 3 test kol ispit</code>
<code>Izlaz: ima</code>	<code>Izlaz: nema</code>

**Zadatak 3.35** Datoteka `studenti.txt` sadrži spisak studenata. Za svakog studenta poznat je nalog na Alas-u (oblika npr. `mr97125`, `mm09001`), ime i prezime i broj poena. Napisati program koji sortira (korišćenjem funkcije `qsort`) studente

### 3 Algoritmi pretrage i sortiranja

---

po broju poena (ukoliko je prisutna opcija `-p`) ili po nalogu (ukoliko je prisutna opcija `-n`). Studenti se po nalogu sortiraju tako što se sortiraju na osnovu godine, zatim na osnovu smeru, i na kraju na osnovu broja indeksa. Ukoliko je u komandnoj liniji uz opciju `-n` naveden i nalog nekog studenta, funkcijom `bsearch` potražiti i prijaviti broj poena studenta sa tim nalogom. Sortirane studente upisati u datoteku `izlaz.txt`.

#### Test 1

<b>Poziv:</b> <code>./a.out -n mm13321</code>	
<b>Datoteka:</b>	<b>Izlaz:</b>
<code>mr14123 Marko Antic 20</code>	<code>mm13321 Marija Radic 12</code>
<code>mm13321 Marija Radic 12</code>	
<code>ml13011 Ivana Mitrovic 19</code>	
<code>ml13066 Pera Simic 15</code>	
<code>mv14003 Jovan Jovanovic 17</code>	

**Zadatak 3.36** Definisana je struktura:

```
typedef struct { int dan; int mesec; int godina; } Datum;
```

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

#### Test 1

<b>Poziv:</b> <code>./a.out datoteka.txt</code>		
<b>Datoteka:</b>	<b>Ulaz:</b>	<b>Izlaz:</b>
<code>1.1.2013</code>	<code>13.12.2016</code>	<code>postoji</code>
<code>13.12.2016</code>	<code>10.5.2015</code>	<code>ne postoji</code>
<code>11.11.2011</code>	<code>5.2.2009</code>	<code>postoji</code>
<code>3.5.2015</code>		
<code>5.2.2009</code>		

**Zadatak 3.37** Za zadatu celobrojnu matricu dimenzije  $n \times m$  napisati funkciju koja vrši sortiranje vrsta matrice, rastuće na osnovu sume elemenata u vrsti. Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unose dimenzije matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz.



## Test 1

```

Ulaz:  Unesite dimenzije matrice:
      3 2
      Unesite elemente matrice:
      6 -5
      -4 3
      2 1
Izlaz: Sortirana matrica je:
      -4 3
      6 -5
      2 1

```

**Zadatak 3.38** Za zadanu kvadratnu matricu dimenzije  $n$  napisati funkciju koja sortira kolone matrice, opadajuće, na osnovu vrednosti prvog elementa u koloni. Napisati program koji testira ovu funkciju. Sa standardnog ulaza se prvo unosi dimenzija matrice, a zatim redom elementi matrice. Rezultujuću matricu ispisati na standardni izlaz.

## 3.4 Rešenja

## Rešenje 3.1

```

1  #include <stdio.h>
   #include <stdlib.h>
3  #include <time.h>
   #define MAX 1000000
5
   /* pri prevodjenju program linkovati sa bibliotekom librt opcijom
7   -lrt zbog funkcije clock_gettime() */
9
   /* Funkcija pretrazuje niz celih brojeva duzine n, trazeci u
   njemu element x. Pretraga se vrši prostom iteracijom kroz
11  niz. Ako se element pronadje funkcija vraca indeks pozicije
   na kojoj je pronadjen. Ovaj indeks je uvek nenegativan. Ako
13  element nije pronadjen u nizu, funkcija vraca -1, kao
   indikator neuspesne pretrage. */
15 int linearna_pretraga(int a[], int n, int x)
   {
17     int i;
       for (i = 0; i < n; i++)
19         if (a[i] == x)
           return i;
21     return -1;
   }
23
   /* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
25     indeks pozicije nadjenog elementa ili -1, ako element nije
       pronadjen */
27 int binarna_pretraga(int a[], int n, int x)
   {

```

### 3 Algoritmi pretrage i sortiranja

```
29  int levi = 0;
    int desni = n - 1;
31  int srednji;
    /* Dokle god je indeks levi levo od indeksa desni */
33  while (levi <= desni) {
    /* Racunamo srednji indeks */
35    srednji = (levi + desni) / 2;
    /* Ako je srednji element veci od x, tada se x mora nalaziti
37    u levoj polovini niza */
    if (x < a[srednji])
39      desni = srednji - 1;
    /* Ako je srednji element manji od x, tada se x mora
41    nalaziti u desnoj polovini niza */
    else if (x > a[srednji])
43      levi = srednji + 1;
    else
45      /* Ako je srednji element jednak x, tada smo pronasli x na
        poziciji srednji */
47      return srednji;
  }
49  /* ako nije pronadjen vratamo -1 */
    return -1;
51 }

53 /* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
    indeks pozicije nadjenog elementa ili -1, ako element nije
55    pronadjen */
    int interpolaciona_pretraga(int a[], int n, int x)
57 {
    int levi = 0;
59    int desni = n - 1;
    int srednji;
61    /* Dokle god je indeks levi levo od indeksa desni... */
    while (levi <= desni) {
63      /* Ako je element manji od pocetnog ili veci od poslednjeg
        clana u delu niza a[levi],...,a[desni] tada nije u tom
65      delu niza. Ova provera je neophodna, da se ne bi dogodilo
        da se prilikom izracunavanja indeksa srednji izadje izvan
67      opsega indeksa [levi,desni] */
        if (x < a[levi] || x > a[desni])
69          return -1;
        /* U suprotnom, x je izmedju a[levi] i a[desni], pa ako su
71      a[levi] i a[desni] jednaki, tada je jasno da je x jednako
        ovim vrednostima, pa vratamo indeks levi (ili indeks
73      desni, sve jedno je).Ova provera je neophodna, zato sto
        bismo inace prilikom izracunavanja srednji imali deljenje
75      nulom. */
        else if (a[levi] == a[desni])
77          return levi;
        /* Racunamo srednji indeks */
79      srednji =
        levi +
81      ((double) (x - a[levi]) / (a[desni] - a[levi])) *
        (desni - levi);
83      /* NAPOMENA: Indeks srednji je uvek izmedju levi i desni,
        ali ce verovatno biti blize trazenoj vrednosti nego da
```

```

85     smo prosto uvek uzimali srednji element. Ovo se moze
86     porediti sa pretragom rečnika: ako neko trazi rec na
87     slovo 'B', sigurno neće da otvori rečnik na polovini, već
      verovatno negde blize pocetku. */
89     /* Ako je srednji element veci od x, tada se x mora nalaziti
      u levoj polovini niza */
91     if (x < a[srednji])
        desni = srednji - 1;
93     /* Ako je srednji element manji od x, tada se x mora
      nalaziti u desnoj polovini niza */
95     else if (x > a[srednji])
        levi = srednji + 1;
97     else
        /* Ako je srednji element jednak x, tada smo pronasli x na
99         poziciji srednji */
        return srednji;
101 }
    /* ako nije pronadjen vracamo -1 */
103 return -1;
}

105 /* Funkcija main */
107 int main(int argc, char **argv)
{
109     int a[MAX];
    int n, i, x;
111     struct timespec time1, time2, time3, time4, time5, time6;
    FILE *f;
113
    /* provera argumenata komandne linije */
115     if (argc != 3) {
        fprintf(stderr,
117             "koriscenje programa: %s dim_niza trazeni_br\n",
                argv[0]);
119         exit(EXIT_FAILURE);
    }
121
    /* Dimenzija niza */
123     n = atoi(argv[1]);
    if (n > MAX || n <= 0) {
125         fprintf(stderr, "Dimenzija niza neodgovarajuca\n");
        exit(EXIT_FAILURE);
127     }

129     /* Broj koji se trazi */
    x = atoi(argv[2]);
131
    /* Elemente niza odredjujemo slucajno, tako da je svaki
133     sledeci veci od prethodnog. srandom() funkcija obezbedjuje
    novi seed za pozivanje random() funkcije. Kako nas niz ne
135     bi uvek isto izgledao seed smo postavili na tekuće vreme u
    sekundama od Nove godine 1970. random()%100 daje brojeve
137     izmedju 0 i 99 */
    srandom(time(NULL));
139     for (i = 0; i < n; i++)
        a[i] = i == 0 ? random() % 100 : a[i - 1] + random() % 100;

```

```
141  /* Linearna pretraga */
143  printf("Linearna pretraga\n");
145  /* Racunamo vreme proteklo od Nove godine 1970 */
145  clock_gettime(CLOCK_REALTIME, &time1);
147  /* Pretrazujemo niz */
147  i = linearna_pretraga(a, n, x);
149  /* Racunamo novo vreme i razlika predstavlja vreme utroseno za
149  lin pretragu */
149  clock_gettime(CLOCK_REALTIME, &time2);
151  if (i == -1)
151      printf("Element nije u nizu\n");
153  else
153      printf("Element je u nizu na poziciji %d\n", i);
155  printf("-----\n");

157  /* Binarna pretraga */
157  printf("Binarna pretraga\n");
159  clock_gettime(CLOCK_REALTIME, &time3);
159  i = binarna_pretraga(a, n, x);
161  clock_gettime(CLOCK_REALTIME, &time4);
161  if (i == -1)
163      printf("Element nije u nizu\n");
163  else
163      printf("Element je u nizu na poziciji %d\n", i);
165  printf("-----\n");

167  /* Interpolaciona pretraga */
169  printf("Interpolaciona pretraga\n");
169  clock_gettime(CLOCK_REALTIME, &time5);
171  i = interpolaciona_pretraga(a, n, x);
171  clock_gettime(CLOCK_REALTIME, &time6);
173  if (i == -1)
173      printf("Element nije u nizu\n");
175  else
175      printf("Element je u nizu na poziciji %d\n", i);
177  printf("-----\n");

179  /* Upisujemo podatke o izvršavanju programa u log fajl */
181  if ((f = fopen("vremena.txt", "a")) == NULL) {
181      fprintf(stderr, "Neuspesno otvaranje log fajla.\n");
183      exit(EXIT_FAILURE);
183  }

185  fprintf(f, "Dimenzija niza od %d elemenata.\n", n);
185  fprintf(f, "\tLinearna pretraga:%10ld ns\n",
187      (time2.tv_sec - time1.tv_sec) * 1000000000 +
187      time2.tv_nsec - time1.tv_nsec);
189  fprintf(f, "\tBinarna: %19ld ns\n",
189      (time4.tv_sec - time3.tv_sec) * 1000000000 +
191      time4.tv_nsec - time3.tv_nsec);
191  fprintf(f, "\tInterpolaciona: %12ld ns\n\n",
193      (time6.tv_sec - time5.tv_sec) * 1000000000 +
193      time6.tv_nsec - time5.tv_nsec);
195  fclose(f);
```

```

197     return 0;
199 }

```

### Rešenje 3.2

```

#include <stdio.h>

2
int lin_pretgraga_rek(int a[], int n, int x)
4 {
    int tmp;
    /* izlaz iz rekurzije */
    if (n <= 0)
        return -1;
    /* ako je prvi element trazeni */
    if (a[0] == x) /* if (a[n-1] == x) */
        return 0; /* return n - 1; */
    /* pretraga ostatka niza */
    tmp = lin_pretgraga_rek(a + 1, n - 1, x);
    return tmp < 0 ? tmp : tmp + 1;
    /* return lin_pretgraga_rek(a, n - 1, x); */
16 }

18 int bin_pretgraga_rek(int a[], int l, int d, int x)
{
    int srednji;
    if (l > d)
        return -1;
    /* srednja pozicija na kojoj se trazi vrednost x */
    srednji = (l + d) / 2;
    /* ako je sredisnji element trazeni */
    if (a[srednji] == x)
        return srednji;
    /* ako je trazeni broj veci od srednjeg, pretrazujemo desnu
       polovinu niza */
    if (a[srednji] < x)
        return bin_pretgraga_rek(a, srednji + 1, d, x);
    /* ako je trazeni broj manji od srednjeg, pretrazujemo levu
       polovinu niza */
    else
        return bin_pretgraga_rek(a, l, srednji - 1, x);
36 }

38
int interp_pretgraga_rek(int a[], int l, int d, int x)
40 {
    int p;
    if (x < a[l] || x > a[d])
        return -1;
    if (a[d] == a[l])
        return l;
    /* pozicija na kojoj se trazi vrednost x */
    p = l + (d - l) * (x - a[l]) / (a[d] - a[l]);
    if (a[p] == x)
        return p;

```

### 3 Algoritmi pretrage i sortiranja

```
50     if (a[p] < x)
51         return interp_pretgraga_rek(a, p + 1, d, x);
52     else
53         return interp_pretgraga_rek(a, l, p - 1, x);
54 }

56 #define MAX 1024

58 int main()
59 {
60     int a[MAX];
61     int x;
62     int i, indeks;

64     /* učitavamo traženi broj */
65     printf("Unesite traženi broj: ");
66     scanf("%d", &x);

68     /* učitavamo elemente niza sve do kraja ulaza - očekujemo da
        korisnik pritisne CTRL+D za naznaku kraja */
70     i = 0;
71     printf("Unesite sortirani niz elemenata: ");
72     while (scanf("%d", &a[i]) == 1) {
73         i++;
74     }

76     printf("Linearna pretraga\n");
77     indeks = lin_pretgraga_rek(a, i, x);
78     if (indeks == -1)
79         printf("Element se ne nalazi u nizu.\n");
80     else
81         printf("Pozicija elementa je %d.\n", indeks);
82
83     printf("Binarna pretraga\n");
84     indeks = bin_pretgraga_rek(a, 0, i - 1, x);
85     if (indeks == -1)
86         printf("Element se ne nalazi u nizu.\n");
87     else
88         printf("Pozicija elementa je %d.\n", indeks);
89
90     printf("Interpolaciona pretraga\n");
91     indeks = interp_pretgraga_rek(a, 0, i - 1, x);
92     if (indeks == -1)
93         printf("Element se ne nalazi u nizu.\n");
94     else
95         printf("Pozicija elementa je %d.\n", indeks);
96
97     return 0;
98 }
```

#### Rešenje 3.3

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

```

5  #define MAX_STUDENATA 128
6  #define MAX_DUZINA 16
7
8  /* 0 svakom studentu imamo 3 informacije i njih objedinjujemo u
9     strukturu kojom cemo predstavljati svakog studenta. */
10 typedef struct {
11     /* indeks mora biti tipa long jer su podaci u datoteci
12        preveliki za int, npr. 20140123 */
13     long indeks;
14     char ime[MAX_DUZINA];
15     char prezime[MAX_DUZINA];
16 } Student;
17
18 /* Ucitani niz studenata ce biti sortirani prema indeksu, jer cemo
19    ih, redom, kako citamo smestati u niz, a u datoteci su vec
20    smesteni sortirani rastuce prema broju indeksa. Iz tog
21    razloga pretragu po indeksu cemo vrsiti binarnom pretragom,
22    dok pretragu po prezimenu moramo vrsiti linearno, jer nemamo
23    garancije da postoji uredjenje po prezimenu. */
24
25 /* Funkcija trazi u sortiranom nizu a[] duzine n broj x. Vraca
26    indeks pozicije nadjenog elementa ili -1, ako element nije
27    pronadjen */
28 int binarna_pretraga(Student a[], int n, long x)
29 {
30     int levi = 0;
31     int desni = n - 1;
32     int srednji;
33     /* Dokle god je indeks levi levo od indeksa desni */
34     while (levi <= desni) {
35         /* Racunamo srednji indeks */
36         srednji = (levi + desni) / 2;
37         /* Ako je srednji element veci od x, tada se x mora nalaziti
38            u levoj polovini niza */
39         if (x < a[srednji].indeks)
40             desni = srednji - 1;
41         /* Ako je srednji element manji od x, tada se x mora
42            nalaziti u desnoj polovini niza */
43         else if (x > a[srednji].indeks)
44             levi = srednji + 1;
45         else
46             /* Ako je srednji element jednak x, tada smo pronasli x na
47                poziciji srednji */
48             return srednji;
49     }
50     /* ako nije pronadjen vratimo -1 */
51     return -1;
52 }
53
54 int linearna_pretraga(Student a[], int n, char x[])
55 {
56     int i;
57     for (i = 0; i < n; i++)
58         /* poredimo prezime i-tog studenta i poslato x */
59         if (strcmp(a[i].prezime, x) == 0)

```

### 3 Algoritmi pretrage i sortiranja

```
        return i;
61     return -1;
    }
63
    /* Main funkcija mora imate argumente jer se ime datoteke dobija
    kao argument komandne linije */
65     int main(int argc, char *argv[])
67     {
        /* Ucitacemo redom sve studente iz datoteke u niz. */
69         Student dosije[MAX_STUDENATA];
        FILE *fin = NULL;
71         int i;
        int br_studenata = 0;
73         long trazen_indeks = 0;
        char trazeno_prezime[MAX_DUZINA];
75
        /* Proveravamo da li nam je korisnik prilikom poziva prosledio
        ime_datoteke sa informacijama o studentima */
77         if (argc != 2) {
79             fprintf(stderr,
                "Greska: Program se poziva sa %s ime_datoteke\n",
81                 argv[0]);
            exit(EXIT_FAILURE);
83         }

85         /* Otvaramo datoteku */
        fin = fopen(argv[1], "r");
87         if (fin == NULL) {
            fprintf(stderr,
89                 "Neuspesno otvaranje datoteke %s za citanje\n",
                argv[1]);
91             exit(EXIT_FAILURE);
        }
93
        /* Citamo sve dok imamo red sa informacijama o studentu */
95         i = 0;
        while (1) {
97             if (i == MAX_STUDENATA)
                break;
99             if (fscanf
                (fin, "%ld %s %s", &dosije[i].indeks, dosije[i].ime,
101                 dosije[i].prezime) != 3)
                break;
103             i++;
        }
105         br_studenata = i;

107         /* Nakon citanja datoteka nam vise nije neophodna i odmah je
        zatvaramo */
109         fclose(fin);

111         printf("Unesite indeks studenta cije informacije zelite: ");
        scanf("%ld", &trazen_indeks);
113         i = binarna_pretraga(dosije, br_studenata, trazen_indeks);
        if (i == -1)
115             printf("Ne postoji student sa indeksom %ld\n",
```



```

        trazen_indeks);
117 else
    printf("Indeks: %ld, Ime i prezime: %s %s\n",
119         dosije[i].indeks, dosije[i].ime, dosije[i].prezime);

121 printf("Unesite prezime studenta cije informacije zelite: ");
    scanf("%s", trazeno_prezime);
123 i = linearna_pretraga(dosije, br_studenata, trazeno_prezime);
    if (i == -1)
125     printf("Ne postoji student sa prezimenom %s\n",
        trazeno_prezime);
127 else
    printf("Indeks: %ld, Ime i prezime: %s %s\n",
129         dosije[i].indeks, dosije[i].ime, dosije[i].prezime);

131 return 0;
}

```

### Rešenje 3.5

```

#include <stdio.h>
2 #include <string.h>
#include <math.h>
4 #include <stdlib.h>

6 /* struktura koja opisuje tacku u ravni */
typedef struct Tacka {
8     float x;
    float y;
10 } Tacka;

12 /* funkcija koja racuna rastojanje zadate tacke od koordinatnog
    pocetka (0,0) */
14 float rastojanje(Tacka A)
{
16     return sqrt(A.x * A.x + A.y * A.y);
}

18 /* funkcija koja pronalazi tacku najblizu koordinatnom pocetku u
    nizu zadatih tacaka t dimenzije n */
20 Tacka najbliza_koordinatnom(Tacka t[], int n)
22 {
    Tacka najbliza;
24     int i;
    najbliza = t[0];
26     for (i = 1; i < n; i++) {
        if (rastojanje(t[i]) < rastojanje(najbliza)) {
28             najbliza = t[i];
        }
30     }
    return najbliza;
32 }

34 /* funkcija koja pronalazi tacku najblizu x osi u nizu zadatih
    tacaka t dimenzije n */

```

### 3 Algoritmi pretrage i sortiranja

---

```
36 Tacka najbliza_x_osi(Tacka t[], int n)
37 {
38     Tacka najbliza;
39     int i;
40     najbliza = t[0];
41     for (i = 1; i < n; i++) {
42         if (fabs(t[i].x) < fabs(najbliza.x)) {
43             najbliza = t[i];
44         }
45     }
46     return najbliza;
47 }
48
49 /* funkcija koja pronalazi tacku najblizu y osi u nizu zadatih
50    tacaka t dimenzije n */
51 Tacka najbliza_y_osi(Tacka t[], int n)
52 {
53     Tacka najbliza;
54     int i;
55     najbliza = t[0];
56     for (i = 1; i < n; i++) {
57         if (fabs(t[i].y) < fabs(najbliza.y)) {
58             najbliza = t[i];
59         }
60     }
61     return najbliza;
62 }
63
64 #define MAX 1024
65
66 int main(int argc, char *argv[])
67 {
68     FILE *ulaz;
69     Tacka tacke[MAX];
70     Tacka najbliza;
71     int i, n;
72
73     /* ocekujemo da korisnik unese barem ime izvrsne verzije
74        programa i ime datoteke sa tackama */
75     if (argc < 2) {
76         fprintf(stderr,
77             "koriscenje programa: %s ime_datoteke\n", argv[0]);
78         return EXIT_FAILURE;
79     }
80
81     /* otvaramo datoteku za citanje */
82     ulaz = fopen(argv[1], "r");
83     if (ulaz == NULL) {
84         fprintf(stderr, "Greska prilikom otvaranja datoteke %s!\n",
85             argv[1]);
86         return EXIT_FAILURE;
87     }
88
89     /* sve dok ima tacaka u datoteci, smestamo ih u niz sa
90        tackama; i predstavlja indeks tekuce tacke */
```

```

92     i = 0;
93     while (fscanf(ulaz, "%f %f", &tacke[i].x, &tacke[i].y) == 2) {
94         i++;
95     }
96     n = i;

97     /* Proveravamo koji su dodatni argumenti komandne linije. Ako
98        nema dodatnih argumenata */
100    if (argc == 2)
101        /* trazimo najblizu tacku u odnosu na koordinatni pocetak */
102        najbliza = najbliza_koordinatnom(tacke, n);
103    /* Inace proveravamo koji je dodatni argument. Ako je u
104       pitanju opcija -x */
105    else if (strcmp(argv[2], "-x") == 0)
106        /* racunamo rastojanje u odnosu na x osu */
107        najbliza = najbliza_x_osi(tacke, n);
108    /* ako je u pitanju opcija -y */
109    else if (strcmp(argv[2], "-y") == 0)
110        /* racunamo rastojanje u odnosu na y osu */
111        najbliza = najbliza_y_osi(tacke, n);
112    else {
113        /* ako nije zadata opcija -x ili -y, ispisujemo obavestenje
114           za korisnika i prekidamo izvršavanje programa */
115        fprintf(stderr, "Pogresna opcija\n");
116        return EXIT_FAILURE;
117    }

118    /* stampamo koordinate trazene tacke */
119    printf("%g %g\n", najbliza.x, najbliza.y);

120    fclose(ulaz);

121    return 0;
122 }

```

### Rešenje 3.6

```

1  #include <stdio.h>
2  #include <math.h>
3
4  /* tacnost */
5  #define EPS 0.001
6
7  int main()
8  {
9      double l, d, s;

10
11     /* posto je u pitanju interval [0, 2] leva granica je 0, a
12        desna 2 */
13     l = 0;
14     d = 2;

15     /* sve dok ne pronadjemo trazenu vrednost argumenta */
16     while (1) {
17         /* pronalazimo sredinu intervala */

```

### 3 Algoritmi pretrage i sortiranja

---

```
19     s = (l + d) / 2;
    /* ako je vrednost kosinusa u ovoj tacki manja od zadate
21     tacnosti, prekidamo pretragu */
    if (fabs(cos(s)) < EPS) {
23         break;
    }
25     /* ako je nula u levom delu intervala, nastavljamo pretragu
    na intervalu [l, s] */
27     if (cos(l) * cos(s) < 0)
        d = s;
29     else
        /* inace, nastavljamo pretragu na intervalu [s, d] */
31     l = s;
    }
33
    /* stampamo vrednost trazene tacke */
35     printf("%g\n", s);
37     return 0;
}
```

#### Rešenje 3.12

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

# Glava 4

## Dinamičke strukture podataka

### 4.1 Liste

**Zadatak 4.1** Napisati program koji koristi jednostruko povezanu listu za čuvanje elemenata koji se unose sa standardnog ulaza. Unošenje novih brojeva u listu prekida se učitavanjem kraja ulaza (EOF). Svako dodavanje novog broja u listu ispratiti ispisivanjem trenutnog sadržaja liste.

- (a) Definirati strukturu `Cvor` koja predstavlja čvor liste.
- (b) Milena: Da li ovde treba dodati i funkciju koja kreira cvor? Nemam resenje kod sebe pa ne znam kako to vec ide, ali mislim da bi trebalo
- (c) Napisati funkciju koja dodaje novi element na početak liste.
- (d) Napisati funkciju koja dodaje novi element na kraj liste.
- (e) Milena: Da li bi ovo trebalo izdvojiti u poseban zadatak? Nekako, ako dodajemo na pocetak i kraj, nemamo garanciju sortiranosti liste, tako da mi to nekako deluje da smo dva zadatka strpali u jedan. Napisati funkciju koja dodaje novi element u listu tako da lista ostane rastuće sortirana.
- (f) Napisati funkciju koja oslobađa memoriju koju je zauzela lista.
- (g) Milena: Ova funkcija bi mogla razlicito da se implementira sa pretpostavkom da je lista sortirana i da nije sortirana, i zato mi dodatno deluje da bi ta dva zadatka trebalo razdvojiti Napisati funkciju koja pretražuje listu za elementom koji ima vrednost koja je argument funkcije.
- (h) Napisati funkciju koja briše sve elemente u listi koji imaju vrednost koja je argument funkcije.
- (i) Milena: Da li ovde nedostaje funkcija koja oslobadja celu memoriju?

Sve funkcije za rad sa listom najpre implementirati iterativno, a zatim i rekursivno.

**Zadatak 4.2** Napisati program koji koristi dvostruko povezanu listu za čuvanje celih brojeva koji se unose sa standardnog ulaza. Unošenje novih brojeva u listu se prekida učitavanjem kraja ulaza (EOF). Svako dodavanje novog broja u listu ispratiti ispisivanjem trenutnog sadržaja liste. **I ovde isto mozda razdvojiti sortiranost od obične liste.**

- (a) Napisati funkciju koja dodaje novi elemenat na početak liste.
- (b) Napisati funkciju koja dodaje novi elemenat na kraj liste.
- (c) Napisati funkciju koja dodaje novi elemenat u listu tako da lista ostane rastuće sortirana.
- (d) Napisati funkciju koja oslobađa memoriju koju je zauzela lista.
- (e) Napisati funkciju koja pretražuje listu za elementom koji ima vrednost koja je argument funkcije.
- (f) Napisati funkciju koja briše sve elemente u listi koji imaju vrednost koja je argument funkcije.

Sve funkcije za rad sa listom implementirati iterativno.

**Zadatak 4.3** Sadržaj datoteke je aritmetički izraz koji može sadržati zagrade {, [ i (. Napisati program koji učitava sadržaj datoteke i korišćenjem steka utvrđuje da li su zagrade u aritmetičkom izrazu dobro uparene. Program štampa odgovarajuću poruku na standardni izlaz. **Milena: promenjeni test primeri, voditi racuna u resenjima sta se stampa!**

### Test 1

```
Datoteka: {[23 + 5344] * (24 - 234)} - 23
Izlaz:   Zagrade su ispravno uparene.
```

### Test 2

```
Datoteka: {[23 + 5] * (9 * 2)} - {23}
Izlaz:   Zagrade su ispravno uparene.
```

### Test 3

```
Datoteka: {[2 + 54] / (24 * 87)} + (234 + 23)
Izlaz:   Zagrade nisu ispravno uparene.
```

**Zadatak 4.4** Napisati program koji proverava ispravnost uparivanja etiketa u HTML datoteci. Ime datoteke se zadaje kao argument komandne linije. **Milena: A sta ako se ne navede argument komandne linije?** Uputstvo: za rešavanje problema koristiti stek implementiran preko listi čiji su čvorovi HTML etikete.

*Test 1*

```

Poziv: ./a.out datoteka.html
Datoteka.html:                                Izlaz:
<html>                                           Ispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
    <h1>Naslov</h1>
    Danas je lep i suncan dan. <br>
    A sutra ce biti jos lepsi.
    <a link="http://www.google.com"> Link 1</a>
    <a link="http://www.math.rs"> Link 2</a>
  </body>
</html>

```

*Test 2*

```

Poziv: ./a.out datoteka.html
Datoteka.html:                                Izlaz:
<html>                                           Neispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
</html>

```

*Test 3*

```

Poziv: ./a.out datoteka.html
Datoteka.html:                                Izlaz:
<html>                                           Neispravno uparene etikete.
  <head><title>Primer</title></head>
  <body>
</body>

```

**Zadatak 4.5** Milena: Problem sa ovim zadatkom je sto je program najpre na usluzi korisnicima, a zatim na usluzi sluzbeniku i to nekako zbunjuje u formulaicji. Formulacija mi nije bila jasna bez citanja resenja, pokusala sam da je preciziran, u nastavku je izmenjena formulacija.

Medjutim, ja i dalje nisam bas zadovoljna i zato predlazem da se formulacija izmeni tako da je program stalno na usluzi sluzbeniku. Program ucitava podatke o prijavljenim korisnicima iz datoteke. Sluzbenik odlucuje da li ce da obradjuje redom korisnike, ili ce u nekim situacijama da odlozi rad sa korisnikom i stavi ga na kraj reda. Program ga uvek pita da na osnovu jmbg-a i zahteva odluci da li ce ga staviti na kraj reda, ako hoce, on ide na kraj reda, ako nece, onda sluzbenik daje odgovor na zahtev i jmbg, zahtev i odgovor se upisuju u izlaznu datoteku.

Napisati program kojim se simulira rad jednog šaltera na kojem se prvo zakazuju termini, a potom službenik uslužuje korisnike redom, kako su se prijavljivali.

Korisnik se prijavljuje unošenjem svog jmbg broja (niska koja sadrži 13 karaktera) i zahteva (niska koja sadrži najviše 999 karaktera). Prijavljivanje korisnika se

prekida unošenjem karaktera za kraj ulaza (EOF).

Službenik redom proziva korisnike čitanjem njihovog `jmbg` broja, a zatim odlučuje da li korisnika vraća na kraj reda ili ga odmah uslužuje. Službeniku se postavlja pitanje `Da li korisnika vracate na kraj reda?` i ukoliko on da odgovor `Da`, korisnik se vraća na kraj reda. Ukoliko odgovor nije `Da`, tada službenik čita korisnikov zahtev. Posle svakog 10 usluženog korisnika, službeniku se nudi mogućnost da prekine sa radom, nevezano od broja korisnika koji i dalje čekaju u redu.

Za čuvanje korisničkih zahteva koristiti red implementiran korišćenjem listi.

**Zadatak 4.6 Milena:** Dodati sta se desava ako nije zadat argument komandne linije ili ako datoteka ne postoji Napisati program koji prebrojava pojavljivanja etiketa HTML datoteke čije se ime zadaje kao argument komandne linije. Rezultat prebrojavanja ispisati na standardni izlaz. Etikete smeštati u listu, a za formiranje liste koristiti strukturu:

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

### Test 1

<b>Poziv:</b> <code>./a.out datoteka.html</code>	
<b>Datoteka.html:</b>	<b>Izlaz:</b>
<code>&lt;html&gt;</code>	<code>a - 4</code>
<code>&lt;head&gt;&lt;title&gt;Primer&lt;/title&gt;&lt;/head&gt;</code>	<code>br - 1</code>
<code>&lt;body&gt;</code>	<code>h1 - 2</code>
<code>&lt;h1&gt;Naslov&lt;/h1&gt;</code>	<code>body - 2</code>
<code>Danas je lep i suncan dan. &lt;br&gt;</code>	<code>title - 2</code>
<code>A sutra ce biti jos lepsi.</code>	<code>head - 2</code>
<code>&lt;a link="http://www.google.com"&gt; Link 1&lt;/a&gt;</code>	<code>html - 2</code>
<code>&lt;a link="http://www.math.rs"&gt; Link 2&lt;/a&gt;</code>	
<code>&lt;/body&gt;</code>	
<code>&lt;/html&gt;</code>	

**Zadatak 4.7 Milena:** i ovde dodati sta ako nema argumenata i ako nema datoteka, kao i u svim ostalim zadacima, a ne bih stalno ovaj komentar ponavljala. Takodje, malo me muci u ovom zadatku sto nema neki smisao. Naime, ako se samo vrsi učitavanje iz datoteka i ispisivanje, onda su ove liste zapravo visak jer isti rezultat moze da se dobije i bez koriscenja listi. Zato mi fali da program uradi nesto sto ne bi mogao da uradi bez koriscenja listi, npr da na osnovu unetog broja ispisuje svaki n-ti broj rezultujuce liste pa to u nekoj petlji da korisnik moze da ispisuje za razlicite unete n ili tako nesto...

Napisati program koji objedinjuje dve sortirane liste. Funkcija ne treba da kreira nove čvorove, već da samo postojeće čvorove preraspodeli. Prva lista se učitava iz datoteke koja se zadaje kao prvi argument komandne linije, a druga iz datoteke čije



se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz.

*Test 1*

```
Poziv: ./a.out dat1.txt dat2.txt
dat1.txt: 2 4 6 10 15
dat2.txt: 5 6 11 12 14 16
Izlaz: 2 4 5 6 6 10 11 12 14 15
      16
```

**Zadatak 4.8** Napisati funkciju koja formira listu studenata tako što se podaci o studentima učitavaju iz datoteke čije se ime zadaje kao argument komandne linije. U svakom redu datoteke nalaze se podaci o studentu i to broj indeksa, ime i prezime. Napisati rekurzivnu funkciju koja određuje da li neki student pripada listi ili ne. Ispisati zatim odgovarajuću poruku i rekurzivno osloboditi memoriju koju je data lista zauzimala. Student se traži na osnovu broja indeksa, koji se zadaje sa standardnog ulaza.

*Test 1*

```
Poziv: ./a.out studenti.txt
Datoteka:      Ulaz:      Izlaz:
123/2014 Marko Lukic      3/2014      da: Ana Sokic
3/2014 Ana Sokic      235/2008      ne
43/2013 Jelena Ilic      41/2009      da: Marija Zaric
41/2009 Marija Zaric
13/2010 Milovan Lazic
```

Milena: Imamo dva zadatka sa labelom 608!

**Zadatak 4.9** Neka su date dve jednostruko povezane liste L1 i L2. Napisati funkciju koja od tih lista formira novu listu L koja sadrži alternirajući raspoređene elemente lista L1 i L2 (prvi element iz L1, prvi element iz L2, drugi element L1, drugi element L2, itd). Ne formirati nove čvorove, već samo postojeće čvorove rasporediti u jednu listu. Prva lista se učitava iz datoteke koja se zadaje kao prvi argument komandne linije, a druga iz datoteke čije se ime zadaje kao drugi argument komandne linije. Rezultujuću listu ispisati na standardni izlaz. Milena: Sta ako je neka lita duza? To precizirati. I ovde me muci sto nedostaje neki smisao zadatku, nesto sto ne bi moglo da se uradi da nismo kristili liste.

*Test 1*

```
Poziv: ./a.out dat1.txt dat2.txt
dat1.txt: 2 4 6 10 15
dat2.txt: 5 6 11 12 14 16
Izlaz: 2 5 4 6 6 11 10 12 15 14
      16
```

## 4 Dinamičke strukture podataka

**Zadatak 4.10** Data je datoteka `brojevi.txt` koja sadrži cele brojeve, po jedan u svakom redu.

- (a) Napisati funkciju koja iz zadate datoteke učitava brojeve i smešta ih u listu.
- (b) Napisati funkciju koja u jednom prolazu kroz zadatu listu celih brojeva pronalazi maksimalan strogo rastući podniz.

Napisati program koji u datoteku `Rezultat.txt` upisuje nađeni strogo rastući podniz. Milena: I ovde me muci sto bi zadatak mogao da se resi i bez koriscenja listi...

Milena: Prirodni oblik testa ovde bi bio horizontalan, a ne ovako vertikalan.

### Test 1

Ulaz: <code>brojevi.txt</code>	Izlaz: <code>Rezultat.txt</code>
43	12
12	15
15	16
16	
4	
2	
8	

**Zadatak 4.11** Grupa od  $n$  plesača na kostimima imaju brojeve od 1 do  $n$ , redom, u smeru kazaljke na satu. Plesači izvode svoju plesnu tačku tako što formiraju krug iz kog najpre izlazi  $k$ -ti plesač. Odbrojavanje se počevši od plesača označenog brojem 1 u smeru kretanja kazaljke na satu. Preostali plesači obrazuju manji krug iz kog opet izlazi  $k$ -ti plesač. Odbrojavanje počinje od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu. Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi  $n$ ,  $k$  ( $k < n$ ) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga. Uputstvo: u implementaciji koristiti kružnu listu.

### Test 1

Ulaz: 5 3
Izlaz: 3 1 5 2 4

Milena: Bilo bi lepo dodati i prethodni zadatak u kojem se smer izbacivanja stalno menja, tako da se onda koristi dvostruko povezana kružna lista.

## 4.2 Stabla

**Zadatak 4.12** Napisati program za rad sa binarnim pretraživačkim stablima.

- (a) Definirati strukturu `Cvor` kojom se opisuje čvor binarnog pretraživačkog stabla koja sadrži ceo broj `b` i pokazivače `levo` i `desno` redom na levo i desno

podstablo<sup>1</sup>.

- (b) Napisati funkciju `Cvor* napravi_cvor(int b)` koja alocira memoriju za novi čvor stabla i vrši njegovu inicijalizaciju zadatim celim brojem `b`.
- (c) Napisati funkciju `void dodaj_u_stablo(Cvor** koren, int b)` koja u stablo na koje pokazuje argument `koren` dodaje ceo broj `b`.
- (d) Napisati funkciju `Cvor* pretrazi_stablo(Cvor* koren, int b)` koja proverava da li se ceo broj `b` nalazi u stablu sa korenom `koren`. Funkcija vraća pokazivač na čvor stabla koji sadrži traženu vrednost ili NULL ukoliko takav čvor ne postoji.
- (e) Napisati funkciju `Cvor* pronadji_najmanji(Cvor* koren)` koja pronalazi čvor koji sadrži najmanju vrednost u stablu sa korenom `koren`.
- (f) Napisati funkciju `Cvor* pronadji_najveci(Cvor* koren)` koja pronalazi čvor koji sadrži najveću vrednost u stablu sa korenom `koren`.
- (g) Napisati funkciju `void obrisi_element(Cvor** koren, int b)` koja briše čvor koji sadrži vrednost `b` iz stabla na koje pokazuje argument `koren`.
- (h) Napisati funkciju `void ispisi_stablo_infiksno(Cvor* koren)` koja infiksno ispisuje sadržaj stabla sa korenom `koren`. Infiksni ispis podrazumeva ispis levog podstabla, korena, a zatim i desnog podstabla.
- (i) Napisati funkciju `void ispisi_stablo_prefiksno(Cvor* koren)` koja prefiksno ispisuje sadržaj stabla sa korenom `koren`. Prefiksni ispis podrazumeva ispis korena, levog podstabla, a zatim i desnog podstabla.
- (j) Napisati funkciju `void ispisi_stablo_postfiksno(Cvor* koren)` koja postfiksno ispisuje sadržaj stabla sa korenom `koren`. Postfiksni ispis podrazumeva ispis levog podstabla, desnog podstabla, a zatim i korena.
- (k) Napisati funkciju `void oslobodi_stablo(Cvor** koren)` koja oslobađa memoriju zauzetu stablom na koje pokazuje argument `koren`.

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

---

<sup>1</sup>U zadacima ove glave u kojima nije eksplicitno naglašen sadržaj čvorova stabla, podrazumevaće se ova struktura.

### Test 1

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

### Test 2

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

**Zadatak 4.13** Napisati program koji izračunava i na standardnom izlazu ispisuje broj pojavljivanja svake reči datoteke čije se ime zadaje kao argument komandne linije. Program realizovati korišćenjem binarnog pretraživačkog stabla uređenog leksikografski prema rečima ne uzimajući u obzir razliku između malih i velikih slova. Ukoliko prilikom pokretanja programa korisnik ne navede ime ulazne datoteke ispisati poruku *Nedostaje ime ulazne datoteke!*.

*Milena: dodati i test primer sa pokretanjem bez ulazne datoteke*

### Test 1

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

Ulazna datoteka test.txt:
Sunce utorak raCunar SUNCE
    programiranje jabuka
    PROGramiranje sunCE JABUka

Izlaz:
jabuka: 2
programiranje: 2
racunar: 1
sunce: 3
utorak: 1
```

### Test 2

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

Ulazna datoteka suma.txt:
lipa zova hrast ZOVA breza LIPA

Izlaz:
breza: 1
hrast: 1
lipa: 2
zova: 2
```

**Zadatak 4.14** U svakoj liniji datoteke čije se ime zadaje sa standardnog ulaza nalazi se ime osobe, prezime osobe i njen broj telefona, npr. *Pera Peric 064/123-4567*. Napisati program koji korišćenjem binarnog pretraživačkog stabla implementira mapu koja sadrži navedene informacije i koja će omogućiti pretragu brojeva telefona za zadata imena i prezimena. Imena i prezimena se unose sve do

unosu reči KRAJ, a za svaki od unetih podataka ispisuje se ili broj telefona ili obaveštenje da traženi broj nije u imeniku. Može se pretpostaviti da imena, prezimena i brojevi telefona neće biti duži od 30 karaktera.

*Test 1*

```

Ulazna datoteka imenik.txt:
  Pera Peric 011/3240-987
  Marko Maric 064/1234-987
  Mirko Maric 011/589-333
  Sanja Savkovic 063/321-098
  Zika Zikic 021/759-858

Unesite ime datoteke: imenik.txt
Unesite ime i prezime: Pera Peric
Broj je: 011/3240-987
Unesite ime i prezime: Marko Markovic
Broj nije u imeniku!
Unesite ime i prezime: KRAJ

```

**Zadatak 4.15** U datoteci `prijemni.txt` nalaze se podaci o prijemnom ispitu učenika jedne osnovne škole tako što je u svakom redu navedeno ime i prezime učenika (niz najviše 50 karaktera), broj poena na osnovu uspeha (realan broj), broj poena na prijemnom ispitu iz matematike (realan broj) i broj poena na prijemnom ispitu iz maternjeg jezika (realan broj). Za učenika koji u zbiru osvoji manje od 10 poena na oba prijemna ispita smatra se da nije položio prijemni. Napisati program koji na osnovu podataka iz ove datoteke formira i prikazuje rang listu učenika. Rang lista sadrži redni broj učenika, njegovo ime i prezime, broj poena na osnovu uspeha, broj poena na prijemnom ispitu iz matematike, broj poena na prijemnom ispitu iz maternjeg jezika i ukupan broj poena i sortirana je opadajuće po ukupnom broju poena. Na rang listi se prvo navode oni učenici koji su položili prijemni ispit, a potom i učenici koji ga nisu položili. Između ovih dveju grupa učenika postoji i horizontalna linija koja ih vizuelno razdvaja.

*Test 1*

```

Datoteka prijemni.txt:
  Marko Markovic 45.4 12.3 11
  Milan Jevremovic 35.2 1.3 9
  Maja Agic 60 19 20
  Nadica Zec 54.2 10 15.8
  Jovana Milic 23.3 2 5.6

Izlaz:
1. Maja Agic 60 19 20 99
2. Nadica Zec 54.2 10 15.8 80
3. Marko Markovic 45.4 12.3 11 68.7
4. Milan Jevremovic 35.2 1.3 9 45.5
-----
5. Jovana Milic 23.3 2 5.6 30.9

```

**Zadatak 4.16** Napisati program koji implementira podsetnik za rođendane. Informacije o rođendanima se nalaze u datoteci čije se ime zadaje kao argument komandne linije u formatu `Ime Prezime DD.MM.YYYY.` - za svaku osobu po jedna linija datoteke. Korisnik unosi datum u naznačenom formatu, a program pronalazi i ispisuje ime i prezime osobe čiji je rođendan zadanog datuma ili ime i prezime osobe koja prva sledeća slavi rođendan. Ovaj postupak treba ponavljati dokle god korisnik ne unese komandu za kraj rada. Informacije o rođendanima uneti u mapu koja je implementirana preko binarnog pretraživačkog stabla i uređena po datumima. Može se pretpostaviti da će svi korišćeni datumi biti validni i u formatu `DD.MM.YYYY.`

### Test 1

```
Ulazna datoteka rodjendani.txt:
Marko Markovic 12.12.1990.
Milan Jevremovic 04.06.1989.
Maja Agic 23.04.2000.
Nadica Zec 01.01.1993.
Jovana Milic 05.05.1990.
...
Unesite datum: 23.04.
Slavljenik: Maja Agic
Unesite datum: 01.01.
Slavljenik: Nadica Zec
Unesite datum: 01.05.
Slavljeni: Jovana Milic 05.05.
Unesite datum: CTRL+D
```

**Zadatak 4.17** Dva binarna stabla su identična ako su ista po strukturi i sadržaju tj. ako oba korena imaju isti sadržaj i identična odgovarajuća podstabla. Napistati funkciju `int identitet(Cvor* koren1, Cvor* koren2)` koja proverava da li su binarna stabla `koren1` i `koren2` koja sadrže cele brojeve identična, a zatim i glavni program koji testira njen rad. Elemente pojedinačnih stabla unositi sa standardnog ulaza sve do pojave broja 0.

### Test 1

```
Ulaz:
Prvo stablo: 10 5 15 3 2 4 30 12 14 13 0
Drugo stablo: 10 15 5 3 4 2 12 14 13 30 0

Izlaz:
Stabla jesu identicna.
```

### Test 2

```
Ulaz:
Prvo stablo: 10 5 15 4 3 2 30 12 14 13 0
Drugo stablo: 10 15 5 3 4 2 12 14 13 30 0

Izlaz:
Stabla nisu identicna.
```

**Zadatak 4.18** Napisati program koji za dva binarna pretraživačka stabla čiji se elementi zadaju sa standardnog ulaza, sve do kraja ulaza, ispisuje uniju, presek i razliku stabla. Unija dva stabala je stablo koje sadrži vrednosti iz oba stabla uračunate tačno po jednom. Presek dva stabala je stablo koje sadrži vrednosti koje se pojavljuju i u prvom i u drugom stablu. Razlika dva stabla je stablo koje sadrži sve vrednosti prvog stabla koje se ne pojavljuju u drugom stablu.

*Test 1*

```

Ulaz:
Prvo stablo: 1 7 8 9 2 2
Drugo stablo: 3 9 6 11 1

Izlaz:
Unija: 1 2 3 6 7 8 9 11
Presek: 1 9
Razlika: 2 7 8

```

**Zadatak 4.19** Napisati funkciju `void sortiraj(int a[], int n)` koja sortira niz celih brojeva `a` dimenzije `n` korišćenjem binarnog pretraživačkog stabla. Napisati i program koji sa standardnog ulaza učitava ceo broj `n` manji od 50 i niz `a` celih brojeva dužine `n`, poziva funkciju `sortiraj` i rezultat ispisuje na standardnom izlazu.

*Test 1*

```

Ulaz:
n: 7
a: 1 11 8 6 37 25 30

Izlaz:
1 6 8 11 25 30 37

```

*Test 2*

```

Ulaz:
n: 55

Izlaz:
Greska: pogresna dimenzija niza!

```

**Zadatak 4.20** Dato je binarno pretraživačko stablo celih brojeva.

- Napisati funkciju koja izračunava broj čvorova stabla.
- Napisati funkciju koja izračunava broj listova stabla.
- Napisati funkciju koja štampa pozitivne vrednosti listova stabla.
- Napisati funkciju koja izračunava zbir čvorova stabla.
- Napisati funkciju koja izračunava najveći element stabla.

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

Napisati program koji testira prethodne funkcije. Stablo formirati na osnovu vrednosti koje se unose sa standardnog ulaza, sve do kraja ulaza, a vrednosti parametara  $i$  i  $x$  pročitati kao argumente komandne linije.

#### Test 2

```
Poziv: ./a.out 2 15
Ulaz:
10 5 15 3 2 4 30 12 14 13
Izlaz:
broj cvorova: 10
broj listova: 4
pozitivni listovi: 2 4 13 30
zbir cvorova: 108
najveci element: 30
dubina stabla: 5
broj cvorova na 2. nivou: 3
elementi na 2. nivou: 3 12 30
maksimalni na 2. nivou: 30
zbir na 2. nivou: 45
zbir elemenata manjih ili jednakih od 15: 7
```

**Zadatak 4.21** Dato je binarno pretraživačko stablo celih brojeva.

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



Napisati program koji testira gorenavedene funkcije. Stablo formirati na osnovu vrednosti koje se unose sa standardnog ulaza, sve do kraja ulaza, a vrednosti parametara  $i$  i  $x$  pročitati kao argumente komandne linije.

*Test 1*

```
Poziv: ./a.out 2 15
Ulaz:
10 5 15 3 2 4 30 12 14 13
Izlaz:
Cvor sa maksimalnim desnim proizvodom: 10
Cvor sa najmanjom levom sumom: 2
Putanja do najdubljeg cvora: 10 15 12 14 13
Putanja do najmanjeg cvora: 10 5 3 2
```

**Zadatak 4.22** Napisati program koji ispisuje sadržaj binarnog pretraživačkog stabla po nivoima.

*Test 1*

```
Ulaz:
10 5 15 3 2 4 30 12 14 13
Izlaz:
0.nivo: 10
1.nivo: 5 15
2.nivo: 3 12 30
3.nivo: 2 4 14
4.nivo: 13
```

**Zadatak 4.23** Dva binarna stabla su *slična kao u ogledalu* ako su ili oba prazna ili ako oba nisu prazna i levo podstablo svakog stabla je *slično kao u ogledalu* desnom podstablu onog drugog (bitna je struktura stabala, ali ne i njihov sadržaj). Napisati funkciju koja proverava da li su dva binarna pretraživačka stabla *slična kao u ogledalu*, a potom i program koji testira rad funkcije nad stablima čiji se elementi unose sa standardnog ulaza sve do unosa broja 0 i to redom za prvo stablo, pa zatim i za drugo stablo.

*Test 1*

```
Ulaz:
Prvo stablo: 11 20 5 3
Drugo stablo: 8 14 30 1
Izlaz:
Stabla su slicna kao u ogledalu.
```

### Test 2

```
Ulaz:
Prvo stablo: 11 20 5 3
Drugo stablo: 8 20 15

Izlaz:
Stabla nisu slicna kao u ogledalu
.
```

**Zadatak 4.24** AVL-stablo je binarno stablo pretrage kod koga apsolutna razlika visina levog i desnog podstabla svakog elementa nije veća od jedan. Napisati funkciju `int avl(Cvor* koren)` koja izračunava broj čvorova stabla sa korenom `koren` koji ispunjavaju uslov za AVL stablo. Napisati zatim i glavni program koji ispisuje rezultat `avl` funkcije za stablo čiji se elementi unose sa standardnog ulaza sve do kraja ulaza.

### Test 1

```
Ulaz:
10 5 15 2 11 16 1 13
Izlaz:
7
```

### Test 2

```
Ulaz:
16 30 40 24 10 18 45 22
Izlaz:
6
```

**Zadatak 4.25** Binarno stablo se naziva HEAP ako je kompletno (svaki njegov čvor, izuzev listova, ima i levog i desnog potomka) i za svaki čvor u stablu važi da je njegova vrednost veća od vrednosti svih ostalih čvorova u njegovim podstablama. Napisati funkciju `int heap(Cvor* koren)` koja proverava da li je dato binarno stablo celih brojeva HEAP. Napisati zatim i glavni program koji ispisuje rezultat `heap` funkcije za stablo čiji se elementi unose sa standardnog ulaza sve do kraja ulaza.

### Test 1

```
Ulaz:
100 19 36 17 3 25 1 2 7
Izlaz:
Da
```

## 4.3 Rešenja

### Rešenje 4.1

```
#include<stdio.h>

2
int main(){
4   printf("Hello bitovi!\n");
```

```
    return 0;
6 }
```

### Rešenje 4.2

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

### Rešenje 4.8

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

### Rešenje 4.4

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

### Rešenje 4.5

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

### Rešenje 4.6

```
#include<stdio.h>
2
int main(){
4     printf("Hello bitovi!\n");
    return 0;
6 }
```

### Rešenje 4.7

```
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello bitovi!\n");
5     return 0;
6 }
```

### Rešenje 4.8

```
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello bitovi!\n");
5     return 0;
6 }
```

### Rešenje 4.9

```
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello bitovi!\n");
5     return 0;
6 }
```

### Rešenje 4.10

```
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello bitovi!\n");
5     return 0;
6 }
```

### Rešenje 4.11

```
1 #include<stdio.h>
2
3 int main(){
4     printf("Hello bitovi!\n");
5     return 0;
6 }
```

### Rešenje 4.12

### Rešenje 4.13

### Rešenje 4.14

Rešenje [4.15](#)

Rešenje [4.16](#)

Rešenje [4.17](#)

Rešenje [4.18](#)

Rešenje [4.19](#)

Rešenje [4.20](#)

Rešenje [4.21](#)

Rešenje [4.22](#)

Rešenje [4.23](#)

Rešenje [4.24](#)

Rešenje [4.25](#)



# Glava 5

## Ispitni rokovi

### 5.1 Programiranje 2, praktični deo ispita, jun 2015.

#### Zadatak 5.1

Kao argument komandne linije zadaje se ime ulazne datoteke u kojoj se nalaze niske. U prvoj liniji datoteke nalazi se informacija o broju niski, a zatim u narednim linijama po jedna niska ne duža od 50 karaktera.

Napisati program u kojem se dinamički alocira memorija za zadati niz niski, a zatim se na standardnom izlazu u redosledu suprotnom od redosleda čitanja ispisuju sve niske koje počinju velikim slovom.

U slučaju pojave bilo kakve greške na standardnom izlazu ispisati vrednost `-1` i prekinuti izvršavanje programa.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre>ž Sadraj datoteke: 5 Programiranje Matematika 12345 dInAmiCnArEc Ispit Izlaz: Ispit Matematika Programiranje</pre>	<pre>ž Sadraj datoteke: 2 maksimalano poena Izlaz:</pre>	<pre>Problem: datoteka ne postoji Izlaz: -1</pre>

#### Zadatak 5.2

Data je biblioteka za rad sa binarnim pretraživačkim stablima čiji čvorovi sadrže cele brojeve. Napisati funkciju `int sumirajN (Cvor * koren, int n)` koja izračunava zbir svih čvorova koji se nalaze na  $n$ -tom nivou stabla (koren se nalazi na nultom nivou, njegova deca na prvom nivou i tako redom). Ispravnost napisane funkcije testirati na osnovu zadate `main` funkcije i biblioteke za rad sa pretraživačkim stablima.

## 5 Ispitni rokovi

Napisati program koji sa standardnog ulaza učitava najpre prirodan broj  $n$ , a potom i brojeve sve do pojave nule koje smešta u stablo i ispisuje rezultat pozivanja funkcije `prebrojN` za broj  $n$  i tako kreirano stablo. U slučaju greške na standardni izlaz za grešku ispisati  $-1$ .

<i>Test 1</i>	<i>Test 2</i>
<pre>Ulaz: 2 8 10 3 6 14 13 7 4 0 Izlaz: 20</pre>	<pre>Ulaz: 0 50 14 5 2 4 56 8 52 7 1 0 Izlaz: 50</pre>

**Zadatak 5.3** Sa standardnog ulaza učitava se broj vrsta i broj kolona celobrojne matrice  $A$ , a zatim i elementi matrice  $A$ . Napisati program koji će ispisati indeks kolone u kojoj se nalazi najviše negativnih elemenata. Ukoliko postoji više takvih kolona, ispisati indeks prve kolone. Može se pretpostaviti da je broj vrsta i broj kolona manji od 50. U slučaju greške ispisati vrednost  $-1$  na standardni izlaz za greške.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<pre>Ulaz: 4 5 1 2 3 4 5 -1 2 -3 4 -5 -5 -4 -3 -2 1 -1 0 0 0 0 Izlaz: 0</pre>	<pre>Ulaz: 2 3 0 0 -5 1 2 -4 Izlaz:</pre>	<pre>Ulaz: -2 Izlaz (na stderr): -1</pre>

## 5.2 Programiranje 2, praktični deo ispita, jul 2015.

### Zadatak 5.4

Napisati program koji kao prvi argument komandne linije prima ime dokumenta u kome treba prebrojati sva pojavljivanja tražene niske (bez preklapanja) koja se navodi kao drugi argument komandne linije (iskoristiti funkciju standardne biblioteke `strstr`). U slučaju bilo kakve greške ispisati  $-1$  na standardni izlaz za greške. Pretpostaviti da linije datoteke neće biti duže od 127 karaktera.

Potpis funkcije `strstr`:

```
char *strstr(const char *haystack, const char *needle);
```

Funkcija traži prvo pojavljivanje podniske `needle` u nisci `haystack`, i vraća pokazivač na početak podniske, ili `NULL` ako podniska nije pronađena.



### Test 1

```

Poziv: ./a.out fajl.txt test
Datoteka: Ovo je test primer.
          U njemu se rec test
          javlja
          vise puta. testtesttest
Izlaz: 5
    
```

### Test 2

```

Poziv: ./a.out
Izlaz (na stderr): -1
    
```

### Test 3

```

Poziv: ./a.out fajl.txt foo
Datoteka: (ne postoji)
Izlaz (na stderr): -1
    
```

### Test 4

```

Poziv: ./a.out fajl.txt .
Datoteka: (prazna)
Izlaz: 0
    
```

### Zadatak 5.5

Na početku datoteke "trouglovi.txt" nalazi se broj trouglova čije su koordinate temena zapisane u nastavku datoteke. Napisati program koji učitva trouglove, i ispisuje ih na standardni izlaz sortirane po površini opadajuće (koristiti Heronov obrazac:  $P = \sqrt{s * (s - a) * (s - b) * (s - c)}$ , gde je  $s$  poluobim trougla). U slučaju bilo kakve greške ispisati -1 na standardni izlaz za greške. Ne praviti nikave pretpostavke o broju trouglova u datoteci, i proveriti da li je datoteka ispravno zadata.

### Test 1

```

Datoteka: 4
           0 0 0 1.2 1 0
           0.3 0.3 0.5 0.5 0.9 1
           -2 0 0 0 0 1
           2 0 2 2 -1 -1
Izlaz:    2 0 2 2 -1 -1
           -2 0 0 0 0 1
           0 0 0 1.2 1 0
           0.3 0.3 0.5 0.5 0.9 1
    
```

### Test 2

```

Datoteka: 3
           1.2 3.2 1.1
           4.3
Izlaz:    -1
    
```

### Test 3

```

Datoteka: (nema datoteke)
Izlaz:    -1
    
```

### Test 4

```

Datoteka: 0
Izlaz:
    
```

**Zadatak 5.6** Data je biblioteka za rad sa binarnim pretraživačkim stablima celih brojeba. Napisati funkciju

```
int f3(Cvor *koren, int n)
```

koja u datom stablu prebrojava čvorove na  $n$ -tom nivou, koji imaju tačno jednog potomka. Pretpostaviti da se koren nalazi na nivou 0. Ispravnost napisane funkcije testirati na osnovu zadate main funkcije i biblioteke za rad sa stablima.

<i>Test 1</i>	<i>Test 2</i>	<i>Test 3</i>
<code>Ulaz:</code> 1 5 3 6 1 4 7 9 <code>Izlaz:</code> 1	<code>Ulaz:</code> 2 5 3 6 1 0 4 7 9 <code>Izlaz:</code> 2	<code>Ulaz:</code> 0 4 2 5 <code>Izlaz:</code> 0
<i>Test 4</i>	<i>Test 5</i>	
<code>Ulaz:</code> 3 <code>Izlaz:</code> 0	<code>Ulaz:</code> -1 4 5 1 7 <code>Izlaz:</code> 0	

### 5.3 Rešenja

#### Rešenje 5.1

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <ctype.h>
4 #define MAX 50
5
6 void greska(){
7     printf("-1\n");
8     exit(EXIT_FAILURE);
9 }
10
11 int main(int argc, char* argv[]){
12
13     FILE* ulaz;
14     char** linije;
15     int i, j, n;
16
17     /* Proveravamo argumente komandne linije.
18     */
19     if(argc!=2){
20         greska();
21     }
22
23     /* Otvaramo datoteku čije ime je navedeno kao argument komandne
24     linije neposredno nakon imena programa koji se poziva. */
25     ulaz=fopen(argv[1], "r");
26     if(ulaz==NULL){
27         greska();
28     }
29
30     /* čitavamo broj linija. */
31     fscanf(ulaz, "%d", &n);
32
33     /* Alociramo memoriju na osnovu čitanog broja linija.*/
34     linije=(char**)malloc(n*sizeof(char*));
```

```

35     if(linije==NULL){
        greska();
    }
37     for(i=0; i<n; i++){
        linije[i]=malloc(MAX*sizeof(char));
39         if(linije[i]==NULL){
            for(j=0; j<i; j++){
41                 free(linije[j]);
            }
            free(linije);
            greska();
43         }
45     }
47     /* čUitavamo svih n linija iz datoteke. */
49     for(i=0; i<n; i++){
        fscanf(ulaz, "%s", linije[i]);
51     }

53     /* Ispisujemo u čodgovarajuem poretku čuitane linije koje
zadovoljavaju kriterijum. */
55     for(i=n-1; i>=0; i--){
        if(isupper(linije[i][0])){
            printf("%s\n", linije[i]);
57         }
    }
59     /* dOslobaamo memoriju koju smo čdinamiki alocirali. */
61     for(i=0; i<n; i++){
        free(linije[i]);
63     }

65     free(linije);

67     /* Zatvaramo datoteku. */
    fclose(ulaz);
69

71     /* šZavravamo sa programom. */
    return 0;
73 }

```

## Rešenje 5.2

```

#include <stdio.h>
2  #include "stabla.h"

4
int sumirajN (Cvor * koren, int n){
6     if(koren==NULL){
        return 0;
8     }

10    if(n==0){
        return koren->broj;
    }
}

```

```
12     }
14     return sumirajN(koren->levo, n-1) + sumirajN(koren->desno, n-1);
16 }
18 int main(){
19     Cvor* koren=NULL;
20     int n;
21     int nivo;
22
23     /* Čitamo vrednost nivoa */
24     scanf("%d", &nivo);
25
26     while(1){
27
28         /* Čitamo broj sa standardnog ulaza */
29         scanf("%d", &n);
30
31         /* Ukoliko je korisnik uneo 0, prekidamo dalje čitanje. */
32         if(n==0){
33             break;
34         }
35
36         /* A ako nije, dodajemo procitani broj u stablo. */
37         dodaj_u_stablo(&koren, n);
38
39     }
40
41     /* Ispisujemo rezultat rada žtraene funkcije */
42     printf("%d\n", sumirajN(koren,nivo));
43
44     /* dOslobaamo memoriju */
45     oslobodi_stablo(&koren);
46
47
48     /* Prekidamo šizvravanje programa */
49     return 0;
50 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "stabla.h"
4
5 Cvor* napravi_cvor(int b ) {
6     Cvor* novi = (Cvor*) malloc(sizeof(Cvor));
7     if( novi == NULL)
8         return NULL;
9
10    /* Inicijalizacija polja novog čvora */
11    novi->broj = b;
12    novi->levo = NULL;
13    novi->desno = NULL;
14
15    return novi;
```

```

17 }
19 void oslobodi_stablo(Cvor** adresa_korena) {
21     /* Prazno stablo i nema šta da se doslobaa */
23     if( *adresa_korena == NULL)
25         return;
27     /* Rekurzivno doslobaamo najpre levo, a onda i desno podstablo*/
29     if( (*adresa_korena)->levo )
31         oslobodi_stablo(&(*adresa_korena)->levo);
33     if( (*adresa_korena)->desno)
35         oslobodi_stablo(&(*adresa_korena)->desno);
37     free(*adresa_korena);
39     *adresa_korena =NULL;
41 }
43 void prover_i_alokaciju( Cvor* novi) {
45     if( novi == NULL) {
47         fprintf(stderr, "Malloc greska za nov cvor!\n");
49         exit(EXIT_FAILURE);
51     }
53 }
55 void dodaj_u_stablo(Cvor** adresa_korena, int broj) {
57     /* Postojece stablo je prazno*/
59     if( *adresa_korena == NULL){
61         Cvor* novi = napravi_cvor(broj);
63         prover_i_alokaciju(novi);
65         *adresa_korena = novi; /* Kreirani čvor novi će biti od
67 sada koren stabla*/
69         return;
71     }
73     /* Brojeve šsmetamo u đureeno binarno stablo, pa
75 ako je broj koji ubacujemo manji od broja koji je u korenu */
77     if( broj < (*adresa_korena)->broj) /* dodajemo u levo
79 podstablo */
81         dodaj_u_stablo(&(*adresa_korena)->levo, broj);
83     /* ako je broj manji ili jednak od broja koji je u korenu stabla
85 , dodajemo nov čvor desno od korena */
87     else
89         dodaj_u_stablo(&(*adresa_korena)->desno, broj);
91 }

```

```

1  #ifndef __STABLA_H__
2  #define __STABLA_H__ 1
3
4  /* Struktura kojom se predstavlja čvor drveta */
5  typedef struct dcvor{
6      int broj;
7      struct dcvor* levo, *desno;
8  } Cvor;
9

```

```
11 /* Funkcija alokira prostor za novi čvor drveta, inicijalizuje polja
    strukture i čvraa čpokaziva na nov čvor */
12 Cvor* napravi_cvor(int b );
13
14 /* đOslobaamo čdinamiki alocirani prostor za stablo
15 * Nakon đoslobaanja se u čpozivajuoj funkciji koren
16 * postavljana NULL, jer je stablo prazno */
17 void oslobodi_stablo(Cvor** adresa_korena);
18
19
20 /* Funkcija proverava da li je novi čvor ispravno alocirani,
21 * i nakon toga prekida program */
22 void prover_i_alokaciju( Cvor* novi);
23
24
25 /* Funkcija dodaje novi čvor u stablo i
26 * žaurira vrednost korena stabla u čpozivajuoj funkciji.
27 */
28 void dodaj_u_stablo(Cvor** adresa_korena, int broj);
29
30 #endif
```

### Rešenje 5.3

```
1 #include <stdio.h>
2 #define MAX 50
3
4
5
6 int main(){
7     int m[MAX][MAX];
8     int v, k;
9     int i, j;
10    int max_broj_negativnih, max_indeks_kolone;
11    int broj_negativnih;
12
13    /* čUitavamo dimenzije matrice */
14    scanf("%d", &v);
15    scanf("%d", &k);
16
17    if(v<0 || v>MAX || k<0 || k>MAX){
18        fprintf(stderr, "-1\n");
19        return 0;
20    }
21
22    /* čUitavamo elemente matrice */
23    for(i=0; i<v; i++){
24        for(j=0; j<k; j++){
25            scanf("%d", &m[i][j]);
26        }
27    }
28
29    /*Pronalazimo kolonu koja žsadri čnajvei broj negativnih
    elemenata */
30    max_indeks_kolone=0;
```

```

max_broj_negativnih=0;
32 for(i=0; i<v; i++){
    if(m[i][0]<0){
34         max_broj_negativnih++;
    }
36 }
38
    for(j=0; j<k; j++){
        broj_negativnih=0;
        for(i=0; i<v; i++){
40             if(m[i][j]<0){
42                 broj_negativnih++;
44             }
            if(broj_negativnih>max_broj_negativnih){
46                 max_indeks_kolone=j;
            }
48         }
50     }

52     /* Ispisujemo žtraeni rezultat */
    printf("%d\n", max_indeks_kolone);
54
56     /* šZavravamo program */
    return 0;
}

```

#### Rešenje 5.4

```

1 #include <stdio.h>
#include <stdlib.h>
3 #include <string.h>
#define MAX 128
5
int main(int argc, char **argv) {
7     FILE *f;
    int brojac = 0;
9     char linija[MAX], *p;

11     if (argc != 3) {
        fprintf(stderr, "-1\n");
13         exit(EXIT_FAILURE);
    }

15
    if ((f = fopen(argv[1], "r")) == NULL) {
17         fprintf(stderr, "-1\n");
        exit(EXIT_FAILURE);
19     }

21     while (fgets(linija, MAX, f) != NULL) {
        p = linija;
23         while (1) {
            p = strstr(p, argv[2]);
25             if (p == NULL)

```

```
27     break;
    brojac++;
    p = p + strlen(argv[2]);
29 }
}
31
    fclose(f);
33
    printf("%d\n", brojac);
35
    return 0;
37 }
```

Rešenje [5.5](#)

Rešenje [5.6](#)