PROGRAMIRANJE 1

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

PROGRAMIRANJE 1
Zbirka zadataka

Beograd
2016.

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

PROGRAMIRANJE 1
Zbirka zadataka

Sadržaj

0.1 Pokazivači

Zadatak 0.1.1 Napisati funkciju koja uređuje svoja dva celobrojna argumenta tako da se u prvom nalazi manji a u drugom veći. Napisati koji učitava dva cela broja i ispisuje rezultat poziva funkcije.

Primer 1

```
Interakcija sa programom:
 Unesite vrednosti promenljivih x i y:  2 5
 Uredjene promenljive: x=2, y=5
```

Primer 2

```
Interakcija sa programom:
 Unesite vrednosti promenljivih x i y:  11 -4
 Uredjene promenljive: x=-4, y=11
```

Zadatak 0.1.2 Napisati funkciju koja za boju datu u *rgb* formatu računa *cmy* format po formulama:
$c = 1 - (r/255)$
$m = 1 - (g/255)$
$y = 1 - (b/255)$
Napisati program koji učitava tri cela broja broja (*rgb* format) i ispisuje rezultat poziva funkcije (*cmy* format). Napomena: *Vrednosti boja u rgb formatu su u opsegu* $[0, 255]$.

Primer 1

```
Interakcija sa programom:
 Unesite boju u rgb formatu:  56 111 24
 c=0.78, m=0.56, y=0.91
```

Primer 2

```
Interakcija sa programom:
 Unesite boju u rgb formatu:  156 -90 5
 Nekorektan unos.
```

Primer 3

```
Interakcija sa programom:
 Unesite boju u rgb formatu:  9 0 237
 c=0.96, m=1.00, y=0.07
```

Primer 4

```
Interakcija sa programom:
 Unesite boju u rgb formatu:  300 11 27
 Nekorektan unos.
```

Zadatak 0.1.3 Napisati funkciju koja za dve prave date svojim koeficijentima pravca i slobodnim članovima određuje njihovu tačku preseka. Funkcija treba da vrati 1 ako se prave seku i 0 ako nemaju tačku preseka (ako su paralelne). Napisati program koji učitava podatke o pravama, poziva napisanu funkciju i ispisuje odgovarajuću poruku.

Primer 1

```
Interakcija sa programom:
 Unesite k i n za prvu pravu:  4 5
 Unesite k i n za drugu pravu:  11 -4
 Prave se seku u tacki (1.29,10.14).
```

Primer 2

```
Interakcija sa programom:
 Unesite k i n za prvu pravu:  0.5 -4.7
 Unesite k i n za drugu pravu:  0.5 9.1
 Prave su paralelne.
```

Zadatak 0.1.4 Napisati funkciju `void modifikacija(char* s, char* t, int* br_modifikacija)` koja na osnovu niske s formira nisku t tako što svako malo slovo zamanjuje velikim. Broj izvršenih modifikacija se čuva u okviru argumenta $br\_modifikacija$. Pretpostaviti da niska s neće biti duža od 20 karaktera. Napisati program koji testira rad napisane funkcije.

Primer 1

```
Interakcija sa programom:
 Unesite nisku:  123abc789XY
 Modifikovana niska je: 123ABC789XY
 Broj modifikacija je: 3
```

Primer 2

```
Interakcija sa programom:
 Unesite nisku:  zimA
 Modifikovana niska je: ZIMA
 Broj modifikacija je: 3
```

Primer 3

```
Interakcija sa programom:
 Unesite nisku:  SNEG
 Modifikovana niska je: SNEG
 Broj modifikacija je: 0
```

Primer 4

```
Interakcija sa programom:
 Unesite nisku:  1234
 Modifikovana niska je: 1234
 Broj modifikacija je: 0
```

Zadatak 0.1.5 Napisati funkciju `void interpunkcija(int* br_tacaka, int* br_zareza)` koja prebrojava tačke i zareze u tekstu koji se unosi sa standardnog ulaza. Napisati program koji testira napisanu funkciju.

Primer 1

```
Interakcija sa programom:
 Unesite tekst:
 a.b.c.d
 a,b,,c,d,e
 Broj tacaka: 3
 Broj zareza: 5
```

Primer 2

```
Interakcija sa programom:
 Unesite tekst:
 .....789.....
 Broj tacaka: 10
 Broj zareza: 0
```

Primer 3

```
Interakcija sa programom:
 Unesite tekst:
 sunce
 Broj tacaka: 0
 Broj zareza: 0
```

Zadatak 0.1.6 Napisati funkciju `void par_nepar(int a[], int n, int parni[], int* pn, int neparni[], int* nn)` koja razbija niz a na niz parnih i niz neparnih brojeva. Pokazivači pn i nn redom treba da sadrže broj elemenata niza parnih tj. niza neparnih elemenata. Pretpostaviti da dužina niza a neće biti veća od 50. Napisati program koji testira napisanu funkciju.

Primer 1

```
Interakcija sa programom:
 Unesite broj elemenata niza:  8
 Unesite elemente niza:
 1 8 9 -7 -16 24 77 4
 Niz parnih brojeva: 8 -16 24 4
 Niz neparnih brojeva: 1 9 -7 77
```

Primer 2

```
Interakcija sa programom:
 Unesite broj elemenata niza:  5
 Unesite elemente niza:
 2 4 6 8 -11
 Niz parnih brojeva: 2 4 6 8
 Niz neparnih brojeva: -11
```

Primer 3

```
Interakcija sa programom:
 Unesite broj elemenata niza:  2
 Unesite elemente niza:
 -15 15
 Niz parnih brojeva:
 Niz neparnih brojeva: -15 15
```

Primer 4

```
Interakcija sa programom:
 Unesite broj elemenata niza:  1
 Unesite elemente niza:
 0
 Niz parnih brojeva: 0
 Niz neparnih brojeva:
```

Zadatak 0.1.7 Napisati funckiju `void min_max(float a[], int n, float* min, float* max)` koja izračunava minimalni i maksimalni element niza a dužine n. Napisati program koji učitava niz realnih brojeva maksimalne dužine 50 i ispisuje vrednosti minimuma i maksimuma zaokruženu na tri decimale.

Primer 1

```
Interakcija sa programom:
 Unesite broj elemenata niza:  5
 Unesite elemente niza:
 24.16 -32.11 999.25 14.25 11
 Minimum: -32.110
 Maksimum: 999.250
```

Primer 2

```
Interakcija sa programom:
 Unesite broj elemenata niza:  4
 Unesite elemente niza:
 -5.126 -18.29 44 29.268
 Minimum: -18.290
 Maksimum: 44.000
```

Primer 3

```
Interakcija sa programom:
 Unesite broj elemenata niza:  1
 Unesite elemente niza:
 4.16
 Minimum: 4.160
 Maksimum: 4.160
```

Primer 4

```
Interakcija sa programom:
 Unesite broj elemenata niza:  3
 Unesite elemente niza:
 7.82 18.989 7.82
 Minimum: 7.820
 Maksimum: 18.989
```

Zadatak 0.1.8 Napisati program koji ispisuje broj navedenih argumenata komandne linije, a zatim i same argumenate i njihove redne brojeve.

Primer 1

```
Pokretanje: ./a.out abcde 123 -5 3.7
Interakcija sa programom:
 Broj argumenata je 5:
 0: ./a.out
 1: abcde
 2: 123
 3: -5
 4: 3.7
```

Primer 2

```
Pokretanje: ./a.out
Interakcija sa programom:
 Broj argumenata je 1:
 0: ./a.out
```

Zadatak 0.1.9 Napisati program koji ispisuje zbir numeričkih argumenata komandne linije. Uputstvo: *Koristiti funkciju* `atoi`.

Primer 1

```
Pokretanje: ./a.out 5 mkp 9 -2 11 a 4 2
Interakcija sa programom:
 Zbir numerickih argumenata: 29
```

Primer 2

```
Pokretanje: ./a.out ab u f hj
Interakcija sa programom:
 Zbir numerickih argumenata: 0
```

Primer 3

```
Pokretanje: ./a.out 33 1 p 44
Interakcija sa programom:
 Zbir numerickih argumenata: 78
```

Primer 4

```
Pokretanje: ./a.out
Interakcija sa programom:
 Zbir numerickih argumenata: 0
```

Zadatak 0.1.10 Napisati program koji ispisuje argumente komandne linije koji počinju slovom *z*.

Primer 1

```
Pokretanje: ./a.out zima jabuka zvezda Zrak
Interakcija sa programom:
 zima zvezda
```

Primer 2

```
Pokretanje: ./a.out bundeva pomorandza
Interakcija sa programom:
```

Primer 3

```
Pokretanje: ./a.out sanke zapad zujanje
Interakcija sa programom:
 zapad zujanje
```

Primer 4

```
Pokretanje: ./a.out
Interakcija sa programom:
```

Zadatak 0.1.11 Napisati program koji ispisuje broj argumenata komandne linije koji sadrže slovo z.

Primer 1

```
Pokretanje: ./a.out zvezda grozd jesen kisa
Interakcija sa programom:
 2
```

Primer 2

```
Pokretanje: ./a.out AZBUKA deda mraz
Interakcija sa programom:
 2
```

Primer 3

```
Pokretanje: ./a.out japan caj
Interakcija sa programom:
 0
```

Primer 4

```
Pokretanje: ./a.out
Interakcija sa programom:
 0
```

Zadatak 0.1.12 Napisati program koji na osnovu broja n koji se zadaje kao argument komandne linije ispisuje cele brojeve iz intervala $[-n,\ n]$.

Primer 1

```
Pokretanje: ./a.out 2
Interakcija sa programom:
 -2 -1 0 1 2
```

Primer 2

```
Pokretanje: ./a.out 4
Interakcija sa programom:
 -4 -3 -2 -1 0 1 2 3 4
```

Primer 3

```
Pokretanje: ./a.out 0
Interakcija sa programom:
 0
```

Primer 4

```
Pokretanje: ./a.out
Interakcija sa programom:
 Greska: nedostaje argument komandne linije!
```

Zadatak 0.1.13 Napisati program koji proverava da li se među zadatim argumentima komandne linije nalaze barem dva ista.

Primer 1

```
Pokretanje: ./a.out pec zima deda mraz pec
Interakcija sa programom:
 Medju argumentima ima istih.
```

Primer 2

```
Pokretanje: ./a.out xyz abc abc abc efgh
Interakcija sa programom:
 Medju argumentima ima istih.
```

Primer 3

```
Pokretanje: ./a.out 11 15 abc 888
Interakcija sa programom:
 Medju argumentima nema istih.
```

Primer 4

```
Pokretanje: ./a.out
Interakcija sa programom:
 Medju argumentima nema istih.
```

Zadatak 0.1.14 Napisati funkciju koja za dva data stringa određuje koliko se uzastopnih karaktera prvog stringa nalazi u drugom stringu počev od početka. Napisati program koji testira napisanu funkciju za dva stringa koji se unose kao argumenti komandne linije.

Primer 1

```
Pokretanje: ./a.out aladin bal
Interakcija sa programom:
 3
```

Primer 2

```
Pokretanje: ./a.out aladin lad
Interakcija sa programom:
 4
```

Primer 3

```
Pokretanje: ./a.out Aladin ala
Interakcija sa programom:
 0
```

Primer 4

```
Pokretanje: ./a.out aladin
Interakcija sa programom:
 Nekorektan poziv
 Program treba pozvati sa ./a.out arg1 arg2
```

Zadatak 0.1.15 Napisati program koji ispisuje sve opcije koje su navedene u komandnoj liniji.

Primer 1

```
Pokretanje: ./a.out -abc input.txt -d -Fg output
Interakcija sa programom:
 a b c d F g
```

Primer 2

```
Pokretanje: ./a.out
Interakcija sa programom:
```

Primer 3

```
Pokretanje: ./a.out ulaz.txt
Interakcija sa programom:
```

Primer 4

```
Pokretanje: ./a.out file.txt -x -yZ -g output
Interakcija sa programom:
 x yZ g
```

Zadatak 0.1.16 Napisati funkciju `void sifruj(char s[], char c, int k)` koja šifruje string s na sledeći način: svako malo i veliko slovo stringa s konvertuje u slovo koje je u abecedi od njega udaljeno k pozicija, i to k pozicija ulevo, ako je karakter c jednak karakteru 'L' ili udesno ako je karakter c jednak karakteru 'D'. Šifrovanje treba da bude kružno. Ako string s sadrži karakter koji nije alfanumerički, ostaviti ga nešifriranog. Napisati program koji testira napisanu funkciju za string i prirodan broj koji se unose kao argumenti komandne linije

dok se pravac šifrovanja unosi kao opcija -p koja može imati vrednosti 'L' ili 'D'. Ukoliko opcija -p nije navedena, podrazumevani pravac je udesno. NAPOMENA: *Možemo podrazumevati da string sadrži najviše 30 karaktera.*

Primer 1

```
POKRETANJE: ./a.out abcd 2
INTERAKCIJA SA PROGRAMOM:
 cdef
```

Primer 2

```
POKRETANJE: ./a.out abcd 2 -p D
INTERAKCIJA SA PROGRAMOM:
 cdef
```

Primer 3

```
POKRETANJE: ./a.out abcd 2 -p L
INTERAKCIJA SA PROGRAMOM:
 yzab
```

Primer 4

```
POKRETANJE: ./a.out abcd -3 -p L
INTERAKCIJA SA PROGRAMOM:
 Nekorektan unos
```

Primer 5

```
POKRETANJE: ./a.out abcd 3 -p X
INTERAKCIJA SA PROGRAMOM:
 Nekorektan unos
```

Primer 6

```
POKRETANJE: ./a.out ab12cd 2 -p D
INTERAKCIJA SA PROGRAMOM:
 Nekorektan unos
```

Zadatak 0.1.17 Tekst Jovana: Ovaj zadatak nema mnogo smisla, predlazem da ga izbacimo.

Zadatak 0.1.18 Jovana: Da li je ovo zadatak iz pokazivaca? Link ka resenju je pogresan, da nije doslo do greske?
Ako su celi brojevi `a` i `b` argumenti komandne linije napraviti niz `A[0] = a, A[1] = a+1, A[2] = a+2, ..., A[b-a] = b` i ispisati ga. Pretpostaviti da je maksimalna dužina niza 200 elemenata. Proveriti da li $a < b$ i $b - a < 200$ i ako ovi uslovi nisu ispunjeni ispisati poruku da je došlo do greške. U slučaju da je dato manje ili više argumenata komandne linije ispisati poruku o grešci.

Primer 1

```
POKRETANJE: ./a.out 34
INTERAKCIJA SA PROGRAMOM:
 greska
```

Primer 2

```
POKRETANJE: ./a.out 12 20
INTERAKCIJA SA PROGRAMOM:
 12 13 14 15 16 17 18 19 20
```

Primer 3

```
POKRETANJE: ./a.out 30 8
INTERAKCIJA SA PROGRAMOM:
 greska
```

Primer 4

```
POKRETANJE: ./a.out -4 -1
INTERAKCIJA SA PROGRAMOM:
 -4 -3 -2 -1
```

Zadatak 0.1.19 Jovana: Da li je ovo zadatak iz pokazivaca? Link ka resenju je pogresan, da nije doslo do greske?

Parametri komandne linije su n, a i b $(a < b)$. Treba popuniti prvih `n` elemenata niza `A` celim slučajnim brojevima koji su između `a` i `b`. Ištampati niz `A` na standarni izlaz. Maksimalan broj elemenata niza `A` je 200. Ukoliko nisu zadati svi argumenti komandne linije ili ne zadovoljavaju potrebna svojstva ispisati poruku o grešci.

0.2 Rešenja

Rešenje 0.1.19

```c
/*
  Napisati funkciju uredi koja uredjuje svoja dva
  celobrojna argumenta tako da se u prvom nalazi manji
  a u drugom veci. Napisati potom glavni program koji
  ucitava dva cela broja i uredjuje njihove vrednosti
  primenom napisane funkcije. Na primer, ako su ucitane
  promenljive x=5 i y=2, njihove vrednosti nakon
  primene funkcije uredi treba da budu x=2 i y=5.
*/

#include <stdio.h>

/*
   Argumenti funkcije uredi_pogresno, promenljive a i b,
   predstavljaju lokalne promenljive za ovu funkciju
   i prestaju da postoje po zavrsetku funkcije. Zbog toga
   se efekti razmene vrednosti promenljivih a i b u slucaju
   da je a>b vide u funkciji, ali se ne vide u glavnom programu.
*/
void uredi_pogresno(int a, int b)
{
  int t;

  if (a>b)
  {
     t = a;
     a = b;
     b = t;
  }
  printf("uredi_pogresno :: a=%d, b=%d\n", a, b);
  printf("uredi_pogresno :: &a=%p, &b=%p\n", &a, &b);
}

/*
   Argumenti funkcije uredi_tacno, promenljive pa i pb,
   takodje su lokalne promenljive za ovu funkciju i
   prestaju da postoje kada se funkcija zavrsi.
```

```
   Njima prosledjujemo adrese promenljivih a i b koje zelimo
   da razmenimo u slucaju da je a>b.

   Promenljivoj a pristupamo preko pokazivacke promenljive
   pa sa *pa i slicno, promenljivoj pb pristupamo sa *pb.

   Vrednosti promenljivih *pa i *pb razmenjujemo kao
   i vrednosti bilo koje dve celobrojne promenljive.

*/
void uredi_tacno(int * pa, int * pb)
{
  int t;
  if (*pa>*pb)
  {
     t = *pa;
     *pa = *pb;
     *pb = t;
  }
  printf("uredi_tacno :: *pa=%d, *pb=%d\n ", *pa, *pb);
  printf("uredi_tacno :: pa=%p, pb=%p\n ", pa, pb);

}
int main()
{
  int a,b;

  printf("Unesi dve celobrojne promenljive:");
  scanf("%d%d",&a,&b);

  printf("main :: a=%d, b=%d\n", a,b);
  printf("main :: &a=%p, &b=%p\n", &a, &b);
  uredi_pogresno(a,b);
  printf("main :: nakon uredi_pogresno, a=%d, b=%d\n", a, b);

  /*
     Funkcija uredi_tacno kao argument ima dve pokazivacke
    promenljive
     (int*,int*). Zbog toga joj je u pozivu funkcije neophodno
    proslediti
     adrese promenljivih koje zelimo da uredimo rastuce, &a i &b.
  */

  uredi_tacno(&a, &b);
  printf("main :: nakon uredi_tacno, a=%d, b=%d\n", a, b);

  return 0;
}
```

Rešenje 0.1.2

```c
/*
  Napisati funkciju koja za boju datu u rgb formatu
  racuna cmy format po formulama:
  C = 1 - ( R / 255 )
  M = 1 - ( G / 255 )
  Y = 1 - ( B / 255 )

  Napisati program koji ucitava boju u rgb formatu,
  primenjuje odgovarajucu funkciju i ispisuje boju u cmy formatu.

*/

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

void rgb_to_cmy(float* a, float* b, float* c)
{
  /* Zagrade su neophodne jer aritmeticke operacije
     imaju veci prioritet od operatora dereferenciranja (*).
  */
  *a=1-(*a)/255;
  *b=1-(*b)/255;
  *c=1-(*c)/255;

  /*
  Pomocu return ne mozemo vratiti vise od jedne vrednosti.

  Ceste greske:
  return a,b,c;         return vraca samo jednu vrednost
  return a; return b; return c; return ce vratiti samo a

  Zato je neophodno da promenljive ciju vrednost
  zelimo da promenimo prenesemo preko pokazivaca.
  */
}


int rgb_korektno(float a)
{
   if(a<0 || a>255)
      return 0;
   return 1;
}


int main()
{
  float a,b,c;

  /*
      Argumenti funkcije rgb_to_cmy su
```

```
      pokazivaci na float. Njima prosledjujemo
      adrese promenljivih a, b i c.
  */

  printf("Unesi boju u rgb formatu (vrednosti izmedju 0 i 255:");
  scanf("%f%f%f",&a,&b,&c);

  if(rgb_korektno(a) && rgb_korektno(b) && rgb_korektno(c))
     rgb_to_cmy(&a,&b,&c);
  else
  {
     printf("Nekorektan unos\n");
     return -1;
  }

  printf("Nakon konverzije: %.2f,%.2f,%.2f\n", a,b,c);

  return 0;
}
```

Rešenje 0.1.3

```
/*
   Napisati funkciju koja za dve prave date svojim koeficijentima
   pravca i slobodnim clanovima odredjuje njihovu tacku preseka.
   Funkcija treba da vrati 1 ako se prave seku i 0 ako nemaju
   tacku preseka (ako su paralelne). Napisati glavni program
   koji ucitava podatke o pravama, poziva napisanu funkciju i
   ispisuje odgovarajucu poruku.
*/

#include<stdio.h>

/*
   Funkcija presek treba da izracuna tri vrednosti:
   1. indikator da li su koeficijenti pravca jednaki ili ne
   2. prvu koordinatu presecne tacke (ukoliko prave nisu paralelne)
   3. drugu koordinatu presecne tacke (ukoliko prave nisu paralelne)

   Indikator funkcija vraca kao povratnu vrednost, preko kljucne reci
   return.

   Koordinate presecne tacke (ako postoji) funkcija vraca preko
   liste argumenata, zbog cega promenljive kojima ce koordinate
   biti dodeljene prenosimo preko pokazivaca (promenljive px i py)

   Promenljive koje sadrze podatke o pravama (k1,n1,k2,n2) se ne
   menjaju u funkciji i zbog toga ih ne moramo prenositi preko
   pokazivaca.
*/
```

```c
int presek(float k1, float n1, float k2, float n2, float* px, float*
    py)
{
   if (k1==k2)
     return 0;

   *px = -(n1-n2)/(k1-k2);
   *py = k1*(*px)+n1;
   return 1;
}

int main()
{
   float k1,k2,n1,n2;
   float x,y;

   printf("Unesi k i n za prvu pravu:");
   scanf("%f%f",&k1,&n1);

   printf("Unesi k i n za drugu pravu:");
   scanf("%f%f",&k2,&n2);

   if(presek(k1,n1,k2,n2,&x,&y))
      printf("Prave se seku u tacki (%.2f,%.2f)\n", x,y);
   else
      printf("Prave su paralelne\n");

   return 0;
}
```

Rešenje 0.1.4

```c
#include <stdio.h>

#define MAX 21

void modifikacija(char *s, char *t, int *br_modifikacija)
{
  int i;
  for(i=0;s[i];i++)
    if(s[i]>='a' && s[i]<='z')
    {
      t[i] = toupper(s[i]);
      (*br_modifikacija)++;
    }
    else
      t[i] = s[i];
}

int main()
{
```

```
  char s[MAX], t[MAX];
  int br_modifikacija = 0;

  printf("Unesite nisku: ");
  scanf("%s", s);

  modifikacija(s, t, &br_modifikacija);

  printf("Modifikovana niska je: %s\nBroj modifikacija je: %d\n", t,
    br_modifikacija);

  return 0;
}
```

Rešenje 0.1.5

```
/*
Napisati funkciju
  void interpunkcija(int * br_tacaka, int * br_zareza)
koja za tekst koji se unosi sa standardnog ulaza sve do kraja ulaza
    prebrojava
broj tacaka i zareza. Napisati zatim program koji testira napisanu
    funkciju.
*/


#include <stdio.h>

void interpunkcija(int* br_tacaka, int* br_zareza){

  int tacke=0, zarezi=0;
  char c;

  while((c=getchar())!=EOF){
    if(c=='.')
      tacke++;

    if(c==',')
      zarezi++;
  }

  *br_tacaka=tacke;
  *br_zareza=zarezi;

}

int main(){
  int br_tacaka, br_zareza;

  printf("Unesite tekst: \n");

```

```
  interpunkcija(&br_tacaka, &br_zareza);

  printf("Broj tacaka: %d\n", br_tacaka);
  printf("Broj zareza: %d\n", br_zareza);

  return 0;
}
```

Rešenje 0.1.6

```
/*
Napisati funkciju
  void par_nepar(int a[], int n, int parni[], int* pn, int neparni[],
     int* nn)
koja razbija niz a na niz parnih i niz neparnih brojeva. Pokazivaci
    pn i nn
redom treba da sadrze broj elemenata niza parnih tj. niza neparnih
    elemenata.
Pretpostaviti da duzina niza a nece biti veca od 50. Napisati program
     koji
testira napisanu funkciju.
*/

#include <stdio.h>
#define MAX 50

void par_nepar(int a[], int n, int parni[], int* pn, int neparni[],
    int* nn){

  int i, j, k;

  /* i - brojac niza a */
  /* j - brojac niza parnih brojeva */
  /* k - brojac niza neparnih brojeva */

  for(i=0, j=0, k=0; i<n; i++){
      /* Ako je element niza paran */
      if(a[i]%2==0){
          /* Smestamo ga u niz parnih brojeva i uvecavamo indeks niza
     j */
          parni[j]=a[i];
          j++;
      }
      else{
          /* Inace, smestamo ga u niz neparnih brojeva i uvecavamo
    indeks niza k */
          neparni[k]=a[i];
          k++;
      }
  }

```

```
  *pn=j;
  *nn=k;

}

int main(){
  int n, i, j, pn, nn;
  int a[MAX], parni[MAX], neparni[MAX];

  /* Ucitavamo dimenziju niza */
  printf("Unesite broj elemenata niza: ");
  scanf("%d", &n);

  if(n<0 || n>MAX){
      printf("Greska: pogresna dimenzija niza!\n");
      return 0;
  }

  /* Ucitavamo elemente niza */
  printf("Unesite elemente niza: ");
  for(i=0; i<n; i++){
    scanf("%d", &a[i]);
  }

  /* Pozivamo funkciju koja razbija zadati niz na niz parnih i niz
    neparnih */
  par_nepar(a, n, parni, &pn, neparni, &nn);


  /* Ispisujemo dobijene nizove */
  printf("Niz parnih brojeva: ");
  for(i=0; i<pn; i++)
    printf("%d ", parni[i]);
  printf("\n");

  printf("Niz neparnih brojeva: ");
  for(i=0; i<nn; i++)
    printf("%d ", neparni[i]);
  printf("\n");

  return 0;
}
```

Rešenje 0.1.7

```
/*
  Napisati funckiju
    void min_max(float a[], int n, float* min, float* max)
  koja izracunava minimalni i maksimalni element niza a duzine n.
  Napisati zatim i program koji ucitava niz realnih brojeva
    maksimalne
```

```c
  duzine 50 i ispisuje vrednosti minimuma i maksimuma na tri decimale
    .

*/

#include<stdio.h>
#define MAX 50

void min_max(float a[], int n, float* min, float* max){

  int i;

  /* Inicijalizujemo vrednosti minimuma i maksimuma */
  *min=a[0];
  *max=a[0];

  /* Obilazimo preostale elemente niza */
  for(i=1; i<n; i++){

    /* Ako je tekuca vrednost veca od maksimalne, azuriramo maksimum
    */
    if(a[i]>*max){
      *max=a[i];
    }

    /* Ako je tekuca vrednost manja od minimalne, azuriramo minimum
    */
    if(a[i]<*min){
      *min=a[i];
    }
  }
}

int main(){
  int i, n;
  float a[MAX], min, max;

  /* Ucitavamo dimenziju niza */
  printf("Unesite broj elemenata niza: ");
  scanf("%d", &n);

  if(n<0 || n>MAX){
    printf("Greska: pogresna dimenzija niza!\n");
    return 0;
  }

  /* Ucitavamo elemente niza */
  printf("Unesite elemente niza:\n");
  for(i=0; i<n; i++){
    scanf("%f", &a[i]);
  }

```

```
  /* Pozivamo funkciju za racunanje maksimuma i minimuma */
  min_max(a, n, &min, &max);

  /* Ispisujemo rezultat */
  printf("Minimum: %.3f\n", min);
  printf("Maksimum: %.3f\n", max);

  return 0;

}
```

Rešenje 0.1.8

```
/*
    Napisati program koji ispisuje broj navedenih argumenata komandne
     linije,
    a zatim i same argumenate i njihove redne brojeve.
*/

#include <stdio.h>

/*
   Argumenti komandne linije cuvaju se u nizu niski pod nazivom
   argv. Svaki element tog niza odgovara jednom argumentu komandne
   linije pri cemu prvi element predstavlja naziv programa koji
   pokrecemo. Celobrojna promenljiva argc predstavlja ukupan
   broj argumenata komandne linije ukljucujuci i argument koji
   odgovara nazivu programa.
*/

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

   printf("Broj argumenata je: %d\n",argc);

   for(i=0; i<argc; i++)
      printf("%d: %s\n",i,argv[i]);

   return 0;
}
```

Rešenje 0.1.9

```
#include <stdio.h>

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

  int i;
```

```
int s = 0;

/* char *argv[] <--- niz niski koje predstavljaju argumente
   navedene iza poziva programa
 int argc     <--- ukupan broj niski (sa sve nazivom programa)
   navedenih prilikom pozivanja

 Ukoliko je program pozvan sa ./a.out 12 abc 6 5 3ab

 argv[0] = "./a.out".
 argv[1] = "12"
 argv[2] = "abc"
 argv[3] = "6"
 argv[4] = "5"
 argv[5] = "3ab"

 argc iznosi 6


 Kako je argv[] po prirodi niz,
 koristimo tzv. brojacku odnosno
 for petlju
 i obradjujemo svaki od argumenata.
*/

/* Funkcija atoi() prihvata nisku,
 i racuna dekadnu vrednost prosledjene niske,
 dokle god se ona moze racunati.
 Na primer, ukoliko je niska "-123",
 atoi() vraca broj -123.
 Ako je, pak, niska "123abc",
 atoi() ce vratiti 123
 (prilikom prve pojave karaktera koji nije cifra, funkcija prekida
  izracunavanje).

 To za posledicu ima da, ukoliko je funkciji
 prosledjeno nesto
 sto se ne moze pretvoriti u broj,
 na primer niska "abcd",
 funkcija atoi() vraca 0.
*/

for(i = 1; i < argc; i++)
  s += atoi(argv[i]); /* Zbog nacina rada funkcije atoi(), mozemo
   je pozvati nad svim argumentima
            komandne linije, i sabrati odgovarajuce dekadne
   vrednosti.
            Ukoliko neki argument i nije broj, to ne predstavlja
   problem
            jer ce u tom slucaju odgovarajuci sabirak biti 0
          */

```

```
    printf("Zbir numerickih argumenata: %d\n", s);

  return 0;
}
```

Rešenje 0.1.10

```
#include <stdio.h>

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

  int i;

  /* Prolazimo for petljom kroz niz argumenata,
   i trazimo one niske ciji je prvi karakter bas 'z'.
   Ukoliko je trenutni argument koji se ispituje
   argv[i],
   kako je on sam po sebi niska,
   do prvog karaktera dolazimo kao i pri dosadasnjem
   radu sa niskama --> argv[i][0]
                       ^
                       |
                  index prvog karaktera u niski argv[i]
  */

  for(i = 1; i < argc; i++)
    if(argv[i][0] == 'z')
      printf("%s ", argv[i]);

  putchar('\n');

  return 0;
}
```

Rešenje 0.1.11

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

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

  int i;
  int br = 0;

  /* Da bismo proverili da li se karakter 'z' (tj. 'Z')
   nalazi u niski argv[i],
   to mozemo uciniti koriscenjem funkcije
   strchr() koja se nalazi u string.h.
```

```
13
    Ukoliko je karakter sadrzan u okviru niske,
15  strchr() vraca pokazivac na taj karakter
    unutar same niske.
17  Inace, ukoliko se karakter ne nalazi u niski,
    funkcija vraca NULL.
19 */

21 for(i = 1; i < argc; i++)
     if(strchr(argv[i], 'z') != NULL || strchr(argv[i], 'Z') != NULL)
23     br++;

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

27 return 0;
}
```

Rešenje 0.1.12

```
1  #include <stdio.h>
   #include <stdlib.h>
3
   int main(int argc, char *argv[])
5  {
     int n,i;
7
     /*
9      Ispisujemo gresku ukoliko nema dovoljno argumenata komandne
       linije.
     */
11   if(argc != 2)
     {
13     printf("Greska: nedostaje argument komandne linije!\n");
       return -1;
15   }

17   /*
       Pretvaramo argument komandne linije koji je string u ceo broj
       koriscenjem funkcije atoi
19   */
     n = atoi(argv[1]);
21   n = abs(n);

23   for(i=(-1)*n;i<=n;i++)
       printf("%d ",i);
25
     return 0;
27 }
```

Rešenje 0.1.13

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

int main(int argc, char *argv[])
{
  int indikator = 0;
  int i,j;
  /*
    Ukoliko imamo samo jedan argument komandne linije,
    ispisujemo da nema istih i zavrsavamo program.
  */
  if(argc < 2)
  {
    printf("Medju argumentima nema istih.\n");
    return -1;
  }

  /*
    Prolazimo kroz niz argumenata i za svaki posebno proverimo
    da li medju ostalima postoji neki koji mu je jednak i ako postoji
    ispisujemo poruku i zavrsavamo program.
    Ako smo izasli iz prve petlje to znaci da nismo pronasli dva ista
     elementa
    i ispisujemo odgovarajucu poruku.
  */
  for(i=0;i<argc;i++)
  {
    for(j=0;j != i && j<argc; j++)
      if(strcmp(argv[i], argv[j]) == 0)
      {
        printf("Medju argumentima ima istih.\n");
        return 0;
      }
  }

  printf("Medju argumentima nema istih.\n");
  return 0;
}
```

Rešenje 0.1.14

```c
/*
   Napisati funkciju koja za dva data stringa str i
   accept odredjuje koliko se uzastopnih karaktera stringa str
   nalazi u stringu accept pocev od pocetka niza str. Napisati
   potom program koji testira napisanu funkciju  za dva stringa
   koji se unose kao argumenti komandne linije. Primeri upotrebe:
```

```
   1:
   ./a.out aladin bal
   3

   2:
   ./a.out aladin lad
   4

   3:
   ./a.out Aladin ala
   0

*/

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

/*
   Funkcija strspn(str,accept) je ugradjena funkcija koja vraca broj karaktera
   stringa str koji se nalaze u stringu accept, pocev od pocetka stringa str.

   Funkcija strspn se nalazi u zaglavlju string.h.

   Funkcija strspn_klon je jedna implementacija funkcije strspn.

   U zadacima cemo uvek koristiti ugradjenu funkciju strspn osim ako u tekstu zadatka
   nije naglaseno da se ona ne sme koristiti. Funkcija strspn_klon sluzi da pokaze na koji
   nacin radi ugradjena funkcija strspn.

   Ugradjena funkcija strspn poziva se na isti nacin kao funkcija strspn_klon:
   strspn(s1,s2)

*/

int strspn_klon(char str[], char accept[])
{
   int br=0;
   int i;

   for(i=0; str[i];i++)
      if(strchr(accept, str[i])!=NULL)
         br++;
      else      /* ako pronadjemo karakter u stringu str koji nije */
         break; /* u stringu accept, prekidamo petlju */

   return br;
}
```

```c
int main(int argc, char* argv[])
{

   int br;

   if(argc<3)
   {
      printf("Nekorektan poziv\nProgram treba pozvati sa ./a.out arg1
     arg2\n");
      return -1;
   }

   br = strspn_klon(argv[1],argv[2]);
   printf("Broj karaktera stringa %s koji se nalaze u stringu %s,
    pocev od pocetka stringa %s: %d\n", argv[1],argv[2],argv[1],br);
   return 0;
}
```

Rešenje 0.1.19

```c
#include <stdio.h>

void suma(int a, int b, int *s);


int main()
{
    int a,b,s;

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

    suma(a,b,&s);

    printf("suma: %d\n",s);

    return 0;
}

void suma(int a, int b, int *s)
{
    *s = a + b;
}
```

Rešenje 0.1.16

```c
/*
   Napisati funkciju void sifruj(char s[], char c, int k) koja
    sifruje
```

```
   string s na sledeci nacin: svako malo i veliko slovo stringa s konvertuje u
   slovo koje je u abecedi od njega udaljeno k pozicija, i to
   k pozicija ulevo, ako je karakter c jednak karakteru 'L' ili udesno
   ako je karakter c jednak karakteru 'D'. Sifrovanje treba da bude kruzno. Ako string
   s sadrzi karakter koji nije alfanumericki, ostaviti ga nesifriranog.

   Napisati potom glavni program koji testira napisanu funkciju za string i prirodan
   broj koji se unose kao argumenti komandne linije dok se pravac sifrovanja unosi
   kao opcija -p koja moze imati vrednosti 'L' ili 'D'. Ukoliko opcija -p nije
   navedena, podrazumevani pravac je udesno.

   Mozemo podrazumevati da string sadrzi najvise 30 karaktera.

   Primeri upotrebe:

   1:
   ./a.out abcd 2
   cdef

   2:
   ./a.out abcd 2 -p D
   cdef

   3:
   ./a.out abcd 2 -p L
   yzab

   4:
   ./a.out abcd -3 -p L
   Nekorektan unos

   5:
   ./a.out abcd 3 -p X
   Nekorektan unos

   6:
   ./a.out ab12cd 2 -p D
   cd12ef

*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX 31
```

```c

void sifruj(char s[], char c, int k)
{
   int i;
   int znak;
   char t;

   /*
      S obzirom da ce korektnost unosa podataka
      biti ispitana pre poziva funkcije, promenljiva
      c ce imati vrednost 'L' ili 'D'.

      Promenljiva znak ima vrednost 1 ili -1
      i sluzi kao pomocna promenljiva u slucaju
      da prilikom sifriranja konvertovani
      karakter izadje iz opsega malih ili velikih slova.

   */
   znak=1;
   if (c=='L')
      znak = -1;


   for(i=0; s[i];i++)
      if(isalpha(s[i]))
      {
         /*
            Promenljiva t predstavlja sifrirani karakter s[i].
            Ako je promenljiva t izvan opsega malih ili velikih slova
    ,
            dodajemo joj ili oduzimamo ukupan broj slova u abecedi
    (26),
            u zavisnosti od pravca sifriranja, kako bismo omogucili
            kruzno sifriranje.
         */
         t = s[i]+znak*k;
         if((islower(s[i]) && (t<'a' || t>'z')) || (isupper(s[i]) &&
    (t<'A' || t>'Z')))
            s[i]=t-znak*26;
         else
            s[i]=t;
      }
}

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

   int k;
   char pravac;
   char rec[MAX];

   /*
```

```
97          Program mozemo pozivati na dva nacina:
            ./a.out abcd 2
99          ili
            ./a.out abcd 2 -p D
101
            Zbog toga, broj argumenata moze biti 3 ili 5.
103     */
105     if (argc!=3 && argc!=5)
        {
107         printf("Nekorektan unos: broj argumenata moze biti 3 ili 5\n");
            return -1;
109     }
111     /*
            Argumenti komandne linije su stringovi. Ako program pokrecemo
113         na sledeci nacin:
            ./a.out abcd 2 -p D
115         to znaci da je argument koji odgovara dvojci u stvari
            string "2". Da bismo string konvertovali u ceo broj,
117         koristimo ugradjenu funkciju atoi iz biblioteke stdlib.h.
        */
119
        k = atoi(argv[2]);
121
        /*
123         Ispitujemo korektnost datih podataka:
        */
125     if (k<=0)
        {
127         printf("Nekorektan unos: broj pozicija mora biti pozitivan ceo
         broj\n");
            return -1;
129     }
131     /* Korektnost unosa je ispitana, sto znaci da
        argc moze biti 3 ili 5 */
133
        if (argc==3) /* Ako je argc 3: */
135         pravac='D';
        else         /* Ako argc nije 3, tada je sigurno 5, jer je */
137     {            /* korektnost unosa ispitana, a unos je korektan
         jedino za argc==3 ili argc==5 */
            /*
139             Ispitujemo korektnost pretposlednjeg argumenta koji mora da
         bude u formatu "-p".
                Ovaj argument je string argv[3]. Njegovom prvom karakteru (
         koji treba
141             da bude '-') pristupamo sa argv[3][0] a drugom sa argv
         [3][1].
            */
143         if (argv[3][0] != '-')
```

```
        {
            printf("Nekorektan unos: pri zadavanju opcija prvi karakter mora biti '-' \n");
            return -1;
        }

        if (argv[3][1]!='p')
        {
            printf("Nekorektan unos: nedozvoljena opcija\n");
            return -1;
        }

        /*
            Nakon argumenta -p sledi argument koji zadaje vrednost ove opcije. To je
            poslednji argument kome pristupamo sa argv[4]. Ovaj argument treba
            da sadrzi samo jedan karakter - 'L' ili 'D' i njemu pristupamo sa
            argv[4][0].
         */
        if(argv[4][0]=='L' || argv[4][0]=='D')
            pravac=argv[4][0];
        else
        {
            printf("Nekorektan unos: pravac moze biti L ili D\n");
            return -1;
        }
    }

    strcpy(rec, argv[1]);
    sifruj(rec,pravac,k);

    printf("Sifrovana rec: %s\n", rec);

    return 0;
}
```

Rešenje 0.1.19

```
#include <stdio.h>

void suma(int a, int b, int *s);


int main()
{
    int a,b,s;

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

```

```
    suma(a,b,&s);

    printf("suma: %d\n",s);

    return 0;
}

void suma(int a, int b, int *s)
{
    *s = a + b;
}
```

Rešenje 0.1.19

```
#include <stdio.h>

void suma(int a, int b, int *s);


int main()
{
    int a,b,s;

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

    suma(a,b,&s);

    printf("suma: %d\n",s);

    return 0;
}

void suma(int a, int b, int *s)
{
    *s = a + b;
}
```

Rešenje 0.1.19

```
#include <stdio.h>

void suma(int a, int b, int *s);


int main()
{
    int a,b,s;

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

```
12      suma(a,b,&s);

14      printf("suma: %d\n",s);

16      return 0;
}
18
void suma(int a, int b, int *s)
20 {
      *s = a + b;
22 }
```