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 Strukture

Zadatak 0.1.1 Definisati strukturu kojom se predstavlja kompleksan broj. Napisati funkcije koje izračunavaju zbir, razliku, proizvod i količnik dva kompleksna broja, a zati i program koji učitava dva kompleksna broja i ispisuje vrednost zbira, razlike, proizvoda i količnika.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite realni i imaginarni deo prvog broja:  1 2
 Unesite realni i imaginarni deo drugog broja:  -2 3
 Zbir: -1.00+5.00*i
 Razlika: 3.00-1.00*i
 Proizvod: -8.00-1.00*i
 Kolicnik: 0.31-0.54*i
```

[Rešenje 0.1.1]

Zadatak 0.1.2 Definisati strukturu kojom se predstavlja razlomak. Napisati funkcije koje izračunavaju zbir i proizvod dva razlomka. Unosi se broj n a potom i n razlomaka sa standarnog ulaza. Ispisati njihov zbir i proizvod na standardni izlaz.

Primer 1

```
Interakcija sa programom:
 Unesi broj razlomaka:  5
 Uneti razlomke:
 1 2
 7 8
 3 4
 5 6
 2 9
 Suma svih razlomaka je 229/72.
 Proizvod svih razlomaka je 35/576.
```

Primer 2

```
Interakcija sa programom:
 Unesi broj razlomaka:  10
 Uneti razlomke:
 4 3
 12 25
 3 8
 1 3
 8 9
 2 3
 5 6
 -24 50
 7 18
 -7 19
 Suma svih razlomaka je 6089/1368.
 Proizvod svih razlomaka je 1568/577125.
```

[Rešenje 0.1.2]

Zadatak 0.1.3 Zimi su prehlade česte i treba unositi više vitamina C. Struktura *Vocka* sadrži ime voćke (nisku maksimalne dužine 20 karaktera) i količinu vitamina C u miligramima (realan broj). Napisati program koji učitava podatke o voćkama sve do unosa reči `KRAJ` i ispisuje ime voćke sa najviše vitamina C. Pretpostaviti da broj voćki neće biti veći od 50. Napomena: *Probati sa testiranjem zadataka pomoću preusmeravanja.*

Primer 1

```
Interakcija sa programom:
 Unesite ime voćke i njenu količinu vitamina C:  jabuka 4.6
 Unesite ime voćke i njenu količinu vitamina C:  limun 51
 Unesite ime voćke i njenu količinu vitamina C:  kivi 92.7
 Unesite ime voćke i njenu količinu vitamina C:  banana 8.7
 Unesite ime voćke i njenu količinu vitamina C:  pomorandza 53.2
 Unesite ime voćke i njenu količinu vitamina C:  KRAJ
 Voce sa najvise C vitamina je: kivi
```

[Rešenje 0.1.3]

Zadatak 0.1.4 Definisati strukturu `Grad` u kojoj se nalazi ime grada (niska dužine 20 karaktera) i prosečna temperatura u toku decembra (realan broj). Napisati program koji učitava imena n ($0 < n < 50$) gradova i njihove prosečne temperature, a zatim ispisuje one gradove koji imaju idealnu temperaturu za klizanje: od 3 do 8 stepeni. *Napomena: probati sa testiranjem zadataka pomoću preusmeravanja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 4
 Unesite grad i temperaturu:  Beograd 7
 Unesite grad i temperaturu:  Uzice 1.5
 Unesite grad i temperaturu:  Subotica 4
 Unesite grad i temperaturu:  Zrenjanin 9
 Gradovi sa idealnom temperaturom za klizanje u decembru:
 Beograd
 Subotica
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj n: 2
 Unesite grad i temperaturu:  Varsava 11
 Unesite grad i temperaturu:  Prag 2
 Gradovi sa idealnom temperaturom za klizanje u decembru:
```

[Rešenje 0.1.4]

Zadatak 0.1.5 Definisati strukturu `ParReci` koja sadrži reč na srpskom jeziku i odgovarajući prevod na engleski jezik. Napisati program koji do kraja ulaza učitava sve parove reči, a potom za rečenicu koja se zadaje u jednoj liniji ispisati prevod. Ako je reč u rečenici nepoznata umesto nje ispisati odgovarajući broj zvezdica. Maksimalna dužina reči je 50 karaktera, ukupan broj parova reči je maksimalno 100, a maksimalna dužina rečenice je 100 karaktera. NAPOMENA: *Probati sa testiranjem zadataka pomoću preusmeravanja.*

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 zima winter
 godina year
 sreca happiness
 programiranje programming
 caj tea
 Unesite recenicu za prevod:
 piti caj zimi je sreca
 **** tea **** ** happiness
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 zima winter
 pas dog
 sreca happiness
 prijatelj friend
 solja cup
 covek man
 Unesite recenicu za prevod:
 pas je covekov najbolji prijatelj
 dog is ******* best friend
```

[Rešenje 0.1.5]

Zadatak 0.1.6 Cenoteka pomaže kupcima da pronađu najpovoljniju cenu za proizvod koji žele da kupe. Napisati program koji učitava najpre broj različitih prodavnica (ceo broj manji od 50) a zatim i podatke o ceni traženog artikla –

zadaje se naziv prodavnice (niske maksimalne dužine 20 karaktera) i cena u toj prodavnici (realan broj). Korisnik zadaje željenu cenu proizvoda, a program ispisuje imena svih onih prodavnica u kojima je cena proizvoda jednaka ili manja od željene. U slučaju greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Uneti broj prodavnica:  5
idea 58.9
maxi 58.2
roda 55.1
tempo 54.5
interex 57.99
Uneti zeljenu cenu:  57.0
Povoljne prodavnice su:
roda
tempo
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Uneti broj prodavnica:  4
dm 43.2
lily 45.99
benu_apoteke 43.99
sephora 50.99
Uneti zeljenu cenu:  47.00
Povoljne prodavnice su:
dm
lily
benu_apoteke
```

[Rešenje 0.1.6]

Zadatak 0.1.7 Statistički zavod Srbije istražuje kako rade obdaništa u Srbiji. Za dato obdanište dobija spisak n dece sa kolonama: pol (`m` ili `z`), broj godina (od 3 do 6) i ocena koju je dete dalo radu obdaništa (od 1 do 5). Maksimalan broj dece u obdaništu je 200. Napisati program koji za decu datog pola i broja godina ispisuje na tri decimale prosečnu ocenu obdaništa. U slučaju neispravnog unosa ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
Uneti broj dece:  5
Uneti podatke za svako dete, pol,
broj godina i ocenu:
m 3 5
z 3 4
m 4 2
m 5 4
m 3 4
Uneti pol i broj godina:  m 3
Prosecna ocena je: 4.500.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
Uneti broj dece:  10
Uneti podatke za svako dete, pol,
broj godina i ocenu:
m 3 5
z 4 4
m 5 4
z 4 3
z 3 2
z 4 5
m 6 5
z 4 4
z 4 5
m 6 3
Uneti pol i broj godina:  z 4
Prosecna ocena je: 4.200.
```

Primer 3

```
Interakcija sa programom:
Uneti broj dece:  15
Uneti podatke za svako dete, pol,
broj godina i ocenu:
m 3 2
z 7 5
Neispravan broj godina.
```

Primer 4

```
Interakcija sa programom:
Uneti broj dece:  2
Uneti podatke za svako dete, pol,
broj godina i ocenu:
m 3 2
z 3 5
Uneti pol i broj godina:  h 5
Neispravan pol.
```

[Rešenje 0.1.7]

Zadatak 0.1.8 Definisati strukturu kojom se opisuje student. Student je zadat svojim imenom i prezimenom (oba su maksimalne dužine 30 karaktera), smerom (R, I, V, N, T, O) i prosečnom ocenom. Napisati program koji učitava podatke o n studenata, zatim učitava smer i ispisuje imena i prezimena onih studenta koji su sa datog smera. Potom ispisati podatke za studenta koji ima najveći prosek. Ako ima više takvih studenata ispisati sve njih. Maksimalan broj studenata je 2000. U slučaju greške ispisati odgovarajuću poruku.

Primer 1

```
Interakcija sa programom:
Uneti broj studenata:  5
Uneti podatke o studentima:
0. student:  Kocic Marija R 9.14
1. student:  Tanja Mratinkovic R 7.88
2. student:  Mihailo Simic N 8.44
3. student:  Milena Medar I 9.14
4. student:  Ljubica Mihic N 9.00
Uneti smer:  R
Studenti sa R smera:
Kocic Marija
Tanja Mratinkovic
--------------------
Svi studenti koji imaju maksimalni prosek:
Kocic Marija, R, 9.14
Milena Medar, I, 9.14
```

Primer 2

```
Interakcija sa programom:
Uneti broj studenata:  4
Uneti podatke o studentima:
0. student:  Djordje Lazarevic N 9.05
1. student:  Minja Peric W 7.70
Nekorektan smer.
```

[Rešenje 0.1.8]

*** Zadatak 0.1.9** Napisati program koji izračunava prosečnu cenu jedne potrošačke korpe. Potrošačka korpa se sastoji od broja kupljenih artikala i niza kupljenih artikala. Svaki artikal određen je svojim nazivom, količinom i cenom. Program treba da učita broj potrošača n (najviše 100), zatim podatke za n potrošačkih korpi i da na osnovu učitanih podataka izračuna prosečnu cenu potrošačke

korpe. Program ispisuje račune svake potrošačke korpe i na kraju ispisuje prosečnu cenu potrošačke korpe. Možemo pretpostaviti da nijedan potrošač neće kupiti više od 20 artikala, kao i da naziv svakog artikla sadrži maksimalno 30 karaktera.

[Rešenje 0.1.9]

Zadatak 0.1.10 Definisati strukturu `Lopta` sa poljima `poluprecnik` (ceo broj u centimetrima) i `boja` (enumeracioni tip koji uključuje plavu, žutu, crvenu i zelenu boju). Zatim učitati informacije o n lopti ($0 < n < 50$) i ispisati ukupnu zapreminu, kao i broj crvenih lopti. U slučaju greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj lopti:  4
 Unesite dalje poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1.lopta:  4 1
 2.lopta:  1 3
 3.lopta:  2 3
 4.lopta:  10 4
 Ukupna zapremina: 4494.57
 Broj crvenih lopti: 2
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj lopti:  8
 Unesite dalje poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1. lopta:  2 1
 2. lopta:  30 3
 3. lopta:  7 3
 4. lopta:  4 1
 5. lopta:  5 2
 6. lopta:  6 2
 7. lopta:  12 3
 8. lopta:  14 2
 Ukupna zapremina: 134996.34
 Ukupno crvenih lopti: 3
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Unesite broj lopti:  8
 Unesite dalje poluprecnike i boje lopti
 (1-plava, 2-zuta, 3-crvena, 4-zelena):
 1. lopta:  1 2
 2. lopta:  2 10
 Nekorektan unos.
```

[Rešenje 0.1.10]

Zadatak 0.1.11 Napisati program za predstavljanje poligona i izračunavanje njegovog obima i dužine stranica.

(a) Definisati tip podataka `TACKA` pogodan za predstavljanje tačke Dekartovske ravni (čije su x i y koordinate podaci tipa `double`).

(b) Definisati funkciju `double rastojanje(TACKA a, TACKA b)` koja izračunava rastojanje između dve tačke.

(c) Definisati funkciju `unsigned ucitaj_poligon(TACKA* tacke, unsigned n)` koja učitava maksimalno n puta po dve vrednosti tipa `double` (koje predstavljaju koordinate temena poligona) i upisuje ih u zadati niz tačaka. Funkcija vraća broj uspešno učitanih tačaka.

(d) Definisati funkciju `double obim(TACKA* poligon, unsigned n)` koja izračunava obim poligona sa n tačaka u zadatom nizu NAPOMENA: *Prilikom računanja obima ne zaboraviti stranicu koja spaja poslednje i prvo teme.*

(e) Definisati funkciju `double maksimalna_stranica(TACKA* poligon, unsigned n)` koja izračunava dužinu najduže stranice poligona sa n tačaka u zadatom nizu.

(f) Napisati funkciju `double povrsina_trougla(TACKA A, TACKA B, TACKA C)` za računanje površine trougla.

(g) Napisati funkciju `double povrsina(TACKA* poligon, unsigned n)` za računanje površine konveksnog poligona. NAPOMENA: *Zadatak se može rešiti korišćenjem funkcije* ***povrsina_trougla****.*

(h) Napisati program koji učitava poligon sa maksimalno N temena ($0 < N \leq 1000$) i za učitani poligon ispisuje na tri decimale obim, dužinu maksimalnu stranice i površinu. Pretpostaviti da je uneseni poligon konveksan. Poligon mora imati barem 3 temena. U slučaju greške ispisati odgovarajuću poruku.

Primer 1

```
INTERAKCIJA SA PROGRAMOM:
 Uneti maksimalan broj tacaka poligona:  10
 0 0
 0 6
 3 3
 Obim poligona je 14.485.
 Duzina maksimalne stranice je 6.000.
 Povrsina poligona je 9.000.
```

Primer 2

```
INTERAKCIJA SA PROGRAMOM:
 Uneti maksimalan broj tacaka poligona:  10
 0 0
 12 0
 13 2
 16 5
 20 10
 18 15
 15 20
 10 20
 8 15
 3 4
 Obim poligona je 63.566.
 Duzina maksimalne stranice je 12.083.
 Povrsina poligona je 247.500.
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Uneti maksimalan broj tacaka poligona:  4
 0 0
 Neispravan broj tacaka poligona.
```

[Rešenje 0.1.11]

*** Zadatak 0.1.12** Sefinisati strukturu `IZRAZ` kojom se opisuje numerički izraz nad celim brojevima koji se sastoji od dva celobrojna operanda, numeričke operacije (sabiranje, oduzimanje, množenje ili celobrojno deljenje) nad celim brojevima.

(a) Napisati funkciju koja ispituje da li je dati izraz korektno zadat i vraća 1 ako jeste a 0 u suprotnom. Podrazumevamo da je izraz korektno zadat ako operacija odgovara $+$, $-$, $*$ ili $/$ i u slučaju deljenja drugi operand je različit od 0.

(b) Napisati funkciju koja za dati izraz odredjuje vrednost izraza.

(c) Napisati funkciju koja učitava dati izraz. Funkcija treba da učita sa standardnog ulaza izlaz koji je zadat prefiksno — prvo operacija, a potom dva operanda. Funkcija vraća 1 ako je učitavanje bilo uspešno, tj. ako je izraz bio korektno zadat ili 0 u suprotnom.

(d) Napisati funkciju koja štampa dati izraz infiksno, u obliku "$operand_1$ *operacija* $operand_2 = vrednost$".

Napisati glavni program koji učitava prirodan broj n, ($n < 1000$) a zatim n izraza u prefiksnoj notaciji. Program treba da ispiše maksimalnu vrednost unetih izraza i sve izraze čija vrednost je manja od polovine maksimalne vrednosti.

Primer 1

```
Interakcija sa programom:
 Uneti broj izraza:  4
 Uneti izraze u prefiksnoj notaciji:
 + 10 4
 - 9 2
 * 11 2
 / 7 3
 Maksimalna vrednost izraza: 22
 Izrazi cija je vrednost manja
 od polovine maksimalne vrednosti:
 9 - 2 = 7
 7 / 3 = 2
```

Primer 2

```
Interakcija sa programom:
 Uneti broj izraza:  10
 Uneti izraze u prefiksnoj notaciji:
 + 10 2
 - -678 34
 * 77 2
 + 1000 -23
 + 102 4
 - 200 23
 / 67 12
 / 1000 2
 * 44 6
 / 13 1
 Maksimalna vrednost izraza: 977
 Izrazi cija je vrednost manja
 od polovine maksimalne vrednosti:
 10 + 2 = 12
 -678 - 34 = -712
 77 * 2 = 154
 102 + 4 = 106
 200 - 23 = 177
 67 / 12 = 5
 44 * 6 = 264
 13 / 1 = 13
```

Primer 3

```
INTERAKCIJA SA PROGRAMOM:
 Uneti broj izraza:  3
 Uneti izraze u prefiksnoj notaciji:
 * 1 2
 / 3 0
 Deljenje nulom!
 Nekorektan unos
```

[Rešenje 0.1.12]

0.2 Rešenja

Rešenje 0.1.1

```c
#include <stdio.h>

/* Struktura koja opisuje kompleksni broj obuhvata polje za realni
 * i polje za imaginarni deo broja.
 */
typedef struct Complex {

  float re;
  float im;
} Complex;

/* Funkcija kojom se izracunava zbir kompleksnih brojeva. */
Complex saberi(Complex *a, Complex *b) {

  Complex c;
  c.re = a->re + b->re;
  c.im = a->im + b->im;
  return c;
}

/* Funkcija kojom se izracunava razlika kompleksnih brojeva. */
Complex oduzmi(Complex *a, Complex *b) {

  Complex c;
  c.re = a->re - b->re;
  c.im = a->im - b->im;
  return c;
}

/* Funkcija kojom se izracunava proizvod kompleksnih brojeva. */
Complex pomnozi(Complex *a, Complex *b) {
```

```
33    Complex c;
      c.re = a->re * b->re - a->im * b->im;
35    c.im = b->re * a->im + a->re * b->im;
      return c;
37  }

39  /* Funkcija kojom se izracunava kolicnik kompleksnih brojeva. */
    Complex podeli(Complex *a, Complex *b) {
41
      Complex c;
43    c.re = (a->re * b->re + a->im * b->im) / (b->re*b->re + b->im*b->im
        );
      c.im = (b->re * a->im - a->re * b->im) / (b->re*b->re + b->im*b->im
        );
45    return c;
    }
47
    int main() {
49
      Complex a, b;
51    Complex c;

53    /* Ucitavamo kompleksne brojeve. */
      printf("Unesite realni i imaginarni deo prvog broja: ");
55    scanf("%f%f", &a.re, &a.im);

57    printf("Unesite realni i imaginarni deo drugog broja: ");
      scanf("%f%f", &b.re, &b.im);
59
      c = saberi(&a, &b);
61     /* Ukoliko je imaginarni deo negativan,
        * njegov zapis vec ukljucuje znak,
63      * te to treba proveriti.
        * Inace, broj je oblika a+b*i.
65      */
      printf("Zbir: %.2f%c%.2f*i\n", c.re, c.im > 0 ? '+' : ' ', c.im);
67
      c = oduzmi(&a, &b);
69    printf("Razlika: %.2f%c%.2f*i\n", c.re, c.im > 0 ? '+' : ' ', c.im)
        ;

71    c = pomnozi(&a, &b);
      printf("Proizvod: %.2f%c%.2f*i\n", c.re, c.im > 0 ? '+' : ' ', c.im
        );
73
      if(b.re != 0 || b.im != 0) {
75      c = podeli(&a, &b);
        printf("Kolicnik: %.2f%c%.2f*i\n", c.re, c.im > 0 ? '+' : ' ', c.
        im);
77    }
      /* U polju kompleksnih brojeva
```

```
79     * nije dozvoljeno deljenje nulom.
       */
81   else
       printf("Kolicnik ne postoji.\n");
83
85   return 0;
}
```

Rešenje 0.1.2

```
1  #include <stdio.h>

3  typedef struct
   {
5    int brojilac;
     int imenilac;
7  }razlomak;

9  int nzd(int a, int b)
   {
11   int pom;

13   if (a < b)
     {
15     pom = a;
       a = b;
17     b = pom;
     }
19
     while(b != 0)
21   {
       pom = a % b;
23     a = b;
       b = pom;
25   }

27   return a;
   }
29
   razlomak zbir(razlomak a, razlomak b)
31 {
     razlomak c;
33   int nzd_razlomka;

35   c.brojilac = a.brojilac * b.imenilac + b.brojilac*a.imenilac;
     c.imenilac = a.imenilac*b.imenilac;
37
     /* Brojilac i imenilac dobijenog zbira se dele najvecim zajednickim
39    * deliocom.
      */
```

```
41    nzd_razlomka = nzd(c.brojilac, c.imenilac);
42
43    c.brojilac = c.brojilac/nzd_razlomka;
44    c.imenilac = c.imenilac/nzd_razlomka;
45
46    return c;
47  }
48
49  razlomak proizvod(razlomak a, razlomak b)
50  {
51    razlomak c;
52    int nzd_razlomka;
53
54    c.brojilac = a.brojilac*b.brojilac;
55    c.imenilac = a.imenilac*b.imenilac;
56
57    /* Brojilac i imenilac dobijenog zbira se dele najvecim zajednickim
58     * deliocom.
59     */
60    nzd_razlomka = nzd(c.brojilac, c.imenilac);
61
62    c.brojilac = c.brojilac/nzd_razlomka;
63    c.imenilac = c.imenilac/nzd_razlomka;
64
65    return c;
66  }
67
68  int main()
69  {
70    int n, i;
71
72    razlomak suma, proizvod_svih, r;
73
74    printf("Unesi broj razlomaka: ");
75    scanf("%d", &n);
76
77
78    suma.brojilac = 0;
79    suma.imenilac = 1;
80
81    proizvod_svih.brojilac = 1;
82    proizvod_svih.imenilac = 1;
83
84    printf("Uneti razlomke:\n");
85    for(i=0; i<n; i++)
86    {
87      scanf("%d%d", &r.brojilac, &r.imenilac);
88
89      suma = zbir(suma, r);
90      proizvod_svih = proizvod(proizvod_svih, r);
91    }
```

```
93    printf("Suma svih razlomaka je %d/%d.\n", suma.brojilac, suma.
        imenilac);
      printf("Proizvod svih razlomaka je %d/%d.\n", proizvod_svih.
        brojilac, proizvod_svih.imenilac);
95
      return 0;
97  }
```

Rešenje 0.1.3

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

4 #define MAX_DUZINA 21
  #define MAX_BR_VOCKI 50
6
  typedef struct vocka
8 {
    char ime[MAX_DUZINA];
10  float vitamin;
  } VOCKA;
12


14 int main()
  {
16  VOCKA vocke[MAX_BR_VOCKI];
    int i = 0, n, max_vocka;
18  char ime[MAX_DUZINA];

20  /*
      Program ucitava podatke o vockama i smesta ih u niz
22    sve dok se ne unese rec KRAJ ili ucita MAX_BR_VOCKI vocki.
    */
24  do
    {
26    printf("Unesite ime vocke i njenu kolicinu vitamina C: ");
      scanf("%s",ime);
28    /*
        Kada se unese rec KRAJ prekida se petlja.
30    */
      if(strcmp(ime, "KRAJ") == 0)
32      break;

34    /*
        Inace ucitava se kolicina vitamina
36      i ta vrednost se smesta u vocku na poziciji "i".
      */
38    strcpy(vocke[i].ime,ime);
      scanf("%f",&vocke[i].vitamin);
40    i++;
    }
```

```
42    while(i<MAX_BR_VOCKI);

44    n = i;

46    /*
        Pretpostavka je da prva vocka ima najvise vitamina.
48      Petljom se prolazi niz vocki i ukoliko se naidje na vocku koja
        ima vise vitamina
        od one koja trenutno ima najvise, azurira se vrednosti maksimalne
         vocke.
50
        Sve vreme se cuva indeks vocke sa najvise vitamina C.
52    */

54    max_vocka = 0;
      for(i=1;i<n;i++)
56      if(vocke[i].vitamin > vocke[max_vocka].vitamin)
        {
58        max_vocka = i;
        }
60
      printf("Voce sa najvise C vitamina je: %s\n", vocke[max_vocka].ime)
        ;
62
      return 0;
64  }
```

Rešenje 0.1.4

```
1   #include <stdio.h>
    #define MAX_DUZINA 20
3   #define MAX_BR_GRADOVA 50

5   typedef struct Grad{
      char ime_grada[MAX_DUZINA+1];
7     float temperatura;
    }Grad;
9


11  int main(){
      int n, i;
13    Grad grad[MAX_BR_GRADOVA];

15    printf("Unesite broj n: ");
      scanf("%d", &n);
17    if(n<0 || n>MAX_BR_GRADOVA){
        printf("Greska: pogresan unos!\n");
19      return 0;
      }
21
      for(i=0; i<n; i++){
```

```
    printf("Unesite grad i temperaturu: ");
    scanf("%s %f", grad[i].ime_grada, &grad[i].temperatura);
  }

  printf("Gradovi sa idealnom temperaturom za klizanje u decembru:\n");
  for(i=0; i<n; i++){
    if(grad[i].temperatura>=3 && grad[i].temperatura<=8){
      printf("%s\n", grad[i].ime_grada);
    }
  }

  return 0;
}
```

Rešenje 0.1.5

```
#include <stdio.h>
#include <string.h>
#define MAX_DUZINA 21
#define MAX_BR_RECI 100

typedef struct ParReci{
  char sr[MAX_DUZINA+1];
  char en[MAX_DUZINA+1];
}ParReci;


/*
  Funkcija koja u recniku koji sadrzi n reci trazi prevod reci rec i upisuje ga u prevod.
  Ukoliko se rec ne nalazi u recniku, prevod se sastoji od zvezdica pri cemu broj zvezdica odgovara
  duzini nepoznate reci.
*/

void pronadji_prevod(ParReci recnik[], int n, char rec[], char prevod[]){
  int i;

  /* Pretrazuje se recnik i trazi se zadata rec. */
  for(i=0; i<n; i++)
  {
    if(strcmp(recnik[i].sr, rec)==0)
    {
      strcpy(prevod, recnik[i].en);
      return;
    }
  }

```

```
  /* Ukoliko rec nije pronadjena, formira se prevod reci koji se
     sastoji od zvezdica. */
  for(i=0; rec[i]; i++){
    prevod[i]='*';
  }
  prevod[i]='\0';
}


int main(){
  ParReci recnik[MAX_BR_RECI];
  int n;
  char sr[MAX_DUZINA+1];
  char en[MAX_DUZINA+1];
  int i, j, k;
  char rec[MAX_DUZINA+1];
  char prevod[MAX_DUZINA+1];
  char c;

  /* Ucitavaju se parovi reci sa standardnog ulaza sve do kraja ulaza
     . */
  i=0;
  while(scanf("%s %s", sr, en)!=EOF){
    if(i==MAX_BR_RECI)
      break;

    strcpy(recnik[i].sr, sr);
    strcpy(recnik[i].en, en);

    i++;
  }
  /* Broj parova reci se cuva u promenljivoj n. */
  n=i;

  printf("Unesite recenicu za prevod: \n");
  do
  {
    /* Ucitava se rec po rec date recenice i pronalazi se njen prevod
    . */
    scanf("%s", rec);

    pronadji_prevod(recnik, n, rec, prevod);
    printf("%s ", prevod);

    /* Ukoliko je karakter iza reci znak za novi red, onda se prekida
     sa unosom, a ako nije
     * ucitava se sledeca recenica.
     */
    c = getchar();

  }while(c != '\n');

```

```
  putchar('\n');

  return 0;
}
```

Rešenje 0.1.6

```
#include <stdio.h>

#define MAX_PRODAVNICA 50
#define DUZINA_RECI 21

typedef struct
{
  char prodavnica[DUZINA_RECI];
  double cena;
}podatak;

int main()
{
  podatak niz[MAX_PRODAVNICA];
  double zeljena;
  int n, i;

  printf("Uneti broj prodavnica: ");
  scanf("%d", &n);

  if (n <=0 || n > MAX_PRODAVNICA)
  {
    printf("Neispravan broj prodavnica.\n");
    return -1;
  }

  for(i=0; i<n; i++)
  {
    scanf("%s%lf", niz[i].prodavnica, &niz[i].cena);

    if (niz[i].cena <= 0)
    {
      printf("Neispravna cena.\n");
      return -1;
    }
  }

  printf("Uneti zeljenu cenu: ");
  scanf("%lf", &zeljena);

  printf("Povoljne prodavnice su:\n");
  for(i=0; i<n; i++)
    if (niz[i].cena <= zeljena)
      printf("%s\n", niz[i].prodavnica);
```

```
45
   return 0;
47 }
```

Rešenje 0.1.7

```
#include <stdio.h>
2
#define MAX_DECE 200
4
typedef struct
6 {
   char pol;
8  int broj_godina;
   int ocena;
10 }dete;

12 int main()
{
14   int n, i, broj_godina;
   dete niz[MAX_DECE];
16   char blanko, pol;
   int suma, broj_dece;
18
   printf("Uneti broj dece: ");
20   scanf("%d", &n);

22   if (n <= 0 || n > MAX_DECE)
   {
24     printf("Neispravan broj dece.\n");
     return -1;
26   }

28   printf("Uneti podatke za svako dete, pol, broj godina i ocenu:\n");
   for(i=0; i<n; i++)
30   {
     scanf("%c%c%d%d", &blanko, &niz[i].pol, &niz[i].broj_godina, &niz
     [i].ocena);
32
     /* Ispitivanje pogresnog unosa. */
34     if (niz[i].pol != 'm' && niz[i].pol != 'z')
     {
36       printf("Neispravan pol.\n");
       return -1;
38     }
     if (niz[i].broj_godina > 6 || niz[i].broj_godina < 3)
40     {
       printf("Neispravan broj godina.\n");
42       return -1;
     }
44     if (niz[i].ocena < 1 || niz[i].ocena > 5)
```

```
    {
      printf("Neispravna ocena.\n");
      return -1;
    }
  }

  printf("Uneti pol i broj godina: ");
  scanf("%c%c%d", &blanko, &pol, &broj_godina);

  /* Ispitivanje ispravnosti unetih podataka. */
  if (pol != 'm' && pol != 'z')
  {
    printf("Neispravan pol.\n");
    return -1;
  }
  if (broj_godina > 6 || broj_godina < 3)
  {
    printf("Neispravan broj godina.\n");
    return -1;
  }

  suma = 0;
  broj_dece = 0;

  for(i=0; i<n; i++)
    if (niz[i].pol == pol && niz[i].broj_godina == broj_godina)
    {
      suma += niz[i].ocena;
      broj_dece++;
    }

  if (broj_dece == 0)
    printf("Ne postoje deca sa takvim karakteristikama.\n");
  else
    printf("Prosecna ocena je: %.3lf.\n", (double)suma/broj_dece);

  return 0;
}
```

Rešenje 0.1.8

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

#define MAXST 2000
#define MAX 31

typedef struct Student
{
  char ime[MAX];
  char prezime[MAX];
```

```
  char smer;
  float prosek;
} STUDENT;

void provera(char smer)
{
  if (smer != 'R' && smer != 'I' && smer != 'V' && smer != 'N' &&
    smer != 'T' && smer != 'O')
  {
    printf("Nekorektan smer.\n");
    exit(EXIT_FAILURE);
  }
}

void ucitaj(STUDENT* s)
{
  scanf("%s",s->ime);

  scanf("%s",s->prezime);

  getchar();
  scanf("%c",&s->smer);

  scanf("%f", &s->prosek);
}

/* II */
/*
  Kada neku promenljivu prenosimo u funkciju kao argument, obicno
  je prenosimo po vrednosti (bez pokazivaca), ako se ona nece menjati
     u funkciji
  ili po adresi (preko pokazivaca), ako ce se njena vrednost
    promeniti u funkciji.

  Prilikom poziva funkcije, za svaki argument funkcije kreira se
    promenljiva
  koja predstavlja lokalnu kopiju argumenta i koja prestaje da
    postoji po zavrsetku
  funkcije. S obzirom da se strukuture sastoje od vise polja,
    zauzimaju
  vise memorije nego nestrukturne promenljive. Zbog toga je za
    njihovo kopiranje
  potrebno vise vremena i vise memorijskih resursa nego za kopiranje
    nestrukturnih
  promenljivih.

  Da bismo ucinili program efikasnijim, korisno je da strukturu uvek
    kao
  argument funkcije prenosimo po adresi (preko pokazivaca), bez
    obzira
  da li ce se struktura u toj funkciji menjati ili ne. Pokazivac na
    strukturu
```

```
  zauzima manje memorije nego sama struktura pa je izrada njegove kopije
  brza a kopija pokazivaca uzima manji memorijski prostor nego kopija strukture.

  Kada prenosimo strukturnu promenljivu u funkciju po adresi (preko pokazivaca), tada
  imamo mogucnost da je u funkciji menjamo. Ukoliko zelimo da onemogucimo promenu,
  uz argument dodajemo kljucnu rec const. Ako pokusamo da promenimo argument
  funkcije prenesen kao const (npr u funkciji ispisi navedemo naredbu s->smer='X';),
  kompajler ce prijaviti gresku. Na ovaj nacin obezbedjujemo da promenljiva
  koju smo preneli po adresi ne da bismo je promenili vec radi povecanja
  efikasnosti programa, ne bude, cak ni slucajno, izmenjena u funkciji.

*/

void ispisi(const STUDENT* s)
{
  printf("%s %s, %c, %.2f\n",s->ime, s->prezime, s->smer, s->prosek);
}


float najveci_prosek(STUDENT studenti[], int n)
{
  float m;
  int i;

  m = studenti[0].prosek;
  for(i=1;i<n;i++)
    if(m<studenti[i].prosek)
       m=studenti[i].prosek;
  return m;
}

/*
   Struktura moze da bude povratna vrednost funkcije.
*/
STUDENT prvi_student_sa_najvecim_prosekom(STUDENT studenti[], int n,
    float m)
{
  STUDENT s;
  int i;
  for(i=0;i<n;i++)
    if(m == studenti[i].prosek)
    {
       /*
```

```
          Na strukture se moze primenjivati
          naredba dodele.
        */
        s = studenti[i];
        break;
    }
  return s;
}


int main()
{
  STUDENT studenti[MAXST];
  int n;
  int i;
  float max_prosek;
  STUDENT student_sa_max_prosekom;
  int indeks;
  char smer;

  printf("Uneti broj studenata: ");
  scanf("%d", &n);

  if (n<0 || n>MAXST)
  {
     printf("Nekorektan unos\n");
     return -1;
  }

  printf("Uneti podatke o studentima:\n");
  for(i=0;i<n;i++)
  {
    printf("%d. student: ", i);
    ucitaj(&studenti[i]);
    provera(studenti[i].smer);
  }

  printf("Uneti smer: ");
  getchar();
  scanf("%c", &smer);

  provera(smer);

  printf("Studenti sa R smera:\n");
  for(i=0;i<n;i++)
     if(studenti[i].smer == smer)
        printf("%s %s\n",studenti[i].ime, studenti[i].prezime);
  printf("----------------------\n");


  /* Stampamo podatke o svim studentima sa
     maksimalnim prosekom.
```

```
  */

  max_prosek = najveci_prosek(studenti, n);
  printf("Svi studenti koji imaju maksimalni prosek:\n");
  for(i=0;i<n;i++)
     if(studenti[i].prosek == max_prosek)
        ispisi(&studenti[i]);


  return 0;
}
```

Rešenje 0.1.9

```
#include <stdio.h>

#define MAXART 20
#define MAXPOT 100
#define MAXNAZIV 31

typedef struct artikal
{
   char naziv[MAXNAZIV];
   int kolicina;
   float cena;
} ARTIKAL;

typedef struct korpa
{
   int br_art;
   ARTIKAL artikli[MAXART];
} KORPA;

/*
   Funkcija ucitaj_artikal ucitava podatke za jedan
   artikal i vraca 1 ako je ucitavanje bilo uspesno
   a 0 u suprotnom. Ucitavanje je neuspesno ukoliko
   kolicina nekog artikla ili njegova cena nisu pozitivni
   brojevi.

   S obzirom da funkcija ucitaj artikal treba da vrati
   dve vrednosti (ucitanu strukturu i indikator uspesnosti),
   strukturu ARTIKAL prenosimo preko pokazivaca a
   indikator uspesnosti vracamo kao povratnu vrednost.

*/

int ucitaj_artikal(ARTIKAL* a)
{

   scanf("%s", a->naziv);
```

```
    scanf("%d", &a->kolicina);

    if (a->kolicina<=0)
    {
        printf("Nekorektan unos za kolicinu artikla: %d\n", a->kolicina
     );
        return 0;
    }

    scanf("%f",&a->cena);
    if (a->cena<0)
    {
        printf("Nekorektan unos za cenu artikla: %f\n", a->cena);
        return 0;
    }

    return 1;
}

/*
    Funkcija izracunaj_racun izracunava racun date
    potrosacke korpe u kojoj su inicijalizovani
    podaci o broju artikala i o svakom pojedinacnom
    artiklu.
*/
float izracunaj_racun(const KORPA* k)
{
    int i;
    float racun=0;
    for(i=0;i<k->br_art;i++)
        racun+=k->artikli[i].kolicina * k->artikli[i].cena;
    return racun;
}

/*
    Pri ucitavanju korpe, zadaje se broj artikala a zatim
    podaci za svaki artikal.

    Funkcija ucitaj_korpu vraca 1 ako je ucitavanje uspesno
    i 0 u suprotnom. Do neuspesnog ucitavanja moze doci
    ako broj artikala u korpi nije pozitivan ili ako dodje
    do neuspesnog ucitavanja nekog artikla.
*/

int ucitaj_korpu(KORPA* k)
{
    int i;
    scanf("%d", &k->br_art);
    if (k->br_art<=0)
    {
        printf("Nekorektan unos za broj artikala: %d\n", k->br_art);
        return 0;
```

```
89      }
        for(i=0; i<k->br_art;i++)
91          if (ucitaj_artikal(&k->artikli[i])==0)
                return 0;
93
        return 1;
95  }

97  /*
        Funkcija ucitaj_niz_korpi ucitava podatke
99      za niz od n potrosackih korpi. Funkcija
        vraca 1 ako je ucitavanje uspesno i 0 ako
101     nije. Ucitavanje je neuspesno ukoliko ne uspe
        ucitavanje jedne od korpi.
103 */

105 int ucitaj_niz_korpi(KORPA korpe[], int n)
107 {
        int i,j;
        for(i=0; i<n; i++)
109         if(ucitaj_korpu(&korpe[i])==0)
                return 0;
111
        return 1;
113
115 }

117 /*
        Funkcija stampaj_racun ispisuje na
        standardni izlaz racun za datu korpu
119     tako sto za svaki artikal ispise
        naziv, cenu i kolicinu i na kraju
121     ukupnu cenu za kupljene artikle.
123 */

125 void stampaj_racun(const KORPA* k)
127 {
        int i,j;
        for(i=0;i<k->br_art;i++)
            printf("\t%s %d %.2f\n", k->artikli[i].naziv, k->artikli[i].
         kolicina, k->artikli[i].cena);
129     printf("------------------------\n");
        printf("\tukupno: %.2f\n", izracunaj_racun(k));
131
133 }

135 /*
        Funkcija stampaj_racune_za_korpe
        ispisuje na standardni izlaz racune
137     za svaku korpu u nizu potrosackih
        korpi
139 */
```

```c

void stampaj_racune_za_korpe(KORPA korpe[], int n)
{
   int i;
   for (i=0;i<n;i++)
   {
      printf("\nKorpa %d:\n",i);
      stampaj_racun(&korpe[i]);
   }
}

/*
   Funkcija prosek racuna prosecnu cenu
   potrosacke korpe za dati niz potrosackih
   korpi
*/
float prosek(KORPA korpe[], int n)
{
   int i;
   float p;

   for(i=0;i<n;i++)
      p+=izracunaj_racun(&korpe[i]);

   return p/n;
}

int main()
{
   int n;
   KORPA korpe[MAXPOT];

   printf("Unesi broj potrosackih korpi:");
   scanf("%d", &n);

   if(n<0 || n>MAXPOT)
   {
      printf("Nekorektan unos broja potrosackih korpi: %d\n",n);
      return -1;
   }

   if (ucitaj_niz_korpi(korpe, n)==0)
      return -1;

   stampaj_racune_za_korpe(korpe,n);
   printf("Prosecna cena potrosacke korpe: %.2f\n", prosek(korpe, n))
    ;

   return 0;
}
```

Rešenje 0.1.10

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

#define MAX 50

typedef struct lopta {
  int poluprecnik;
  enum {plava, zuta, crvena, zelena} boja;
} LOPTA;

float zapremina(LOPTA l) {
 return pow(l.poluprecnik, 3)*4/3*M_PI;
}


float ukupna_zapremina(LOPTA lopte[], int n) {

  int i;
  float z = 0;

  for(i = 0; i < n; i++)
    z += zapremina(lopte[i]);

  return z;
}

/*
  Funkcija je opstija od trazene i broji sve lopte odredjene boje u
    nizu lopti.
  U zavisnosti od prosledjene boje funkciji, funkcija vraca
    odgovarajuci broj.
*/
int broj_lopti_u_boji(LOPTA lopte[], int n, int boja) {

  int br = 0;
  int i;
  for(i = 0; i < n; i++)
    if(lopte[i].boja == boja)
      br++;
  return br;
}

int main() {

  LOPTA lopte[MAX];
  int n;
  int i;
  int boja;

  printf("Unesite broj lopti: ");
```

```
  scanf("%d", &n);

  if(n < 1 || n > MAX) {

    printf("Nekorektan unos.\n");
    return 0;
  }

  printf("Unesite dalje poluprecnike i boje lopti (1-plava, 2-zuta,
    3-crvena, 4-zelena):\n");
  for(i = 0; i < n; i++) {

    printf("%d. lopta: ", i+1);
    scanf("%d%d", &lopte[i].poluprecnik, &boja);

  /* U zavisnosti od unetog celog broja,
     bira se boja lopte.
   */
    switch(boja) {

      case 1: lopte[i].boja = plava; break;
      case 2: lopte[i].boja = zuta; break;
      case 3: lopte[i].boja = crvena; break;
      case 4: lopte[i].boja = zelena; break;
      default:
        printf("Nekorektan unos.\n");
      return 0;
    }
  }

  printf("Ukupna zapremina: %.2f\n", ukupna_zapremina(lopte, n));

  printf("Ukupno crvenih lopti: %d\n", broj_lopti_u_boji(lopte, n,
    crvena));

  return 0;
}
```

Rešenje 0.1.11

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

#define MAX_TACAKA 1000

typedef struct
{
  int x, y;
}TACKA;

double rastojanje(TACKA a, TACKA b)
```

```
{
  return sqrt(pow(a.x - b.x, 2) + pow(a.y - b.y, 2));
}

unsigned ucitaj_poligon(TACKA* tacke, unsigned n)
{
  int i = 0;

  while(i < n && scanf("%d%d", &tacke[i].x, &tacke[i].y) != EOF)
    i++;

  return i;
}

double obim(TACKA* poligon, unsigned n)
{
  double o = rastojanje(poligon[0], poligon[n-1]);
  int i;

  for(i=0; i<n-1; i++)
    o += rastojanje(poligon[i], poligon[i+1]);

  return o;
}

double maksimalna_stranica(TACKA* poligon, unsigned n)
{
  double max = rastojanje(poligon[0], poligon[n-1]);
  double stranica;
  int i;

  for(i=0; i<n-1; i++)
  {
    stranica = rastojanje(poligon[i], poligon[i+1]);
    if (stranica > max)
      max = stranica;
  }

  return max;
}

double povrsina_trougla(TACKA A, TACKA B, TACKA C)
{
  double a = rastojanje(B, C);
  double b = rastojanje(A, C);
  double c = rastojanje(A, B);

  double s = (a + b + c)/2;

  return sqrt(s*(s -a)*(s - b)*(s - c));
}
```

```
64 double povrsina(TACKA* poligon, unsigned n)
   {
66   double P = 0;
     int i;
68
     for(i=1; i<n-1; i++)
70     P += povrsina_trougla(poligon[0], poligon[i], poligon[i+1]);

72   return P;
   }
74
   int main()
76 {
     int N;
78   unsigned m;
     TACKA poligon[MAX_TACAKA];
80
     printf("Uneti maksimalan broj tacaka poligona: ");
82   scanf("%d", &N);

84   if (N < 3 || N > MAX_TACAKA)
     {
86     printf("Neispravan broj tacaka poligona.\n");
       return -1;
88   }

90   m = ucitaj_poligon(poligon, N);

92   if (m < 3)
     {
94     printf("Neispravan broj tacaka poligona.\n");
       return -1;
96   }

98   printf("Obim poligona je %.3lf.\n", obim(poligon, m));
     printf("Duzina maksimalne stranice je %.3lf.\n",
       maksimalna_stranica(poligon, m));
100  printf("Povrsina poligona je %.3lf.\n", povrsina(poligon, m));

102  return 0;
   }
```

Rešenje 0.1.12

```
1  #include <stdio.h>
   #define MAX 1000
3
   typedef struct
5  {
     char o;
7    int x;
```

```
  int y;
} IZRAZ;


int korektan_izraz(const IZRAZ izraz)
{
   if(izraz.o != '+' && izraz.o != '-' && izraz.o != '*' && izraz.o
    != '/')
   {
      printf("Nedozvoljena operacija!\n");
      return 0;
   }
   if(izraz.o == '/' && izraz.y == 0)
   {
      printf("Deljenje nulom!\n");
      return 0;
   }
   return 1;
}

/* Racunanje vrednosti izraza. */
int vrednost(const IZRAZ izraz)
{
   int v;

   switch (izraz.o)
   {
      case '+':
         return izraz.x + izraz.y;
      case '-':
         return izraz.x - izraz.y;
      case '*':
         return izraz.x * izraz.y;
      case '/':
         return izraz.x / izraz.y;
   }
}


/*
   Promenljiva izraz ce se promeniti u funkciji
   ucitaj_izraz tako sto ce njenim neinicijalizovanim
   poljima o,x,y biti dodeljene vrednosti ucitane
   sa ulaza. Zbog toga ovu promenljivu
   funkciji prosledjujemo po adresi, preko pokazivaca.

   S obzirom da ucitavanje karaktera nije prvo
   ucitavanje koje se obavlja u programu, funkcijom
   getchar() se ucita karakter kojim se razdvaja
   unos karaktera od prethodnog unosa (najcesce blanko
   znak ili znak za novi red).
```

```
59 */
60
61 int ucitaj_izraz(IZRAZ* izraz)
62 {
63    getchar();
64    scanf("%c%d%d",&izraz->o, &izraz->x, &izraz->y);
65    if (!korektan_izraz(*izraz))
66       return 0;
67    return 1;
68 }
69
70
71 void stampaj_izraz(const IZRAZ izraz)
72 {
73    printf("%d %c %d = %d\n", izraz.x, izraz.o, izraz.y, vrednost(
74     izraz));
75 }
76
77 int max_vr(IZRAZ izrazi[], int n)
78 {
79    int i;
80    int max;
81
82    max=vrednost(izrazi[0]);
83
84    for(i=1; i<n; i++)
85       if(vrednost(izrazi[i])>max)
86          max=vrednost(izrazi[i]);
87
88    return max;
89 }
90
91 int main()
92 {
93    int n;
94    IZRAZ izrazi[MAX];
95    int max;
96    int i;
97
98    printf("Uneti broj izraza: ");
99    scanf("%d", &n);
100   if(n<0 || n>MAX)
101   {
102      printf("Nekorektna vrednost broja n!\n");
103      return -1;
104   }
105
106
107   printf("Uneti izraze u prefiksnoj notaciji:\n");
108   for(i=0; i<n; i++)
109      if(ucitaj_izraz(&izrazi[i])==0)
110      {
```

```
            printf("Nekorektan unos\n");
            return -1;
        }


    max = max_vr(izrazi, n);
    printf("Maksimalna vrednost izraza:%d\n", max);

    printf("Izrazi cija je vrednost manja od polovine maksimalne
     vrednosti:\n");

    for(i=0; i<n; i++)
        if(vrednost(izrazi[i])<max/2)
            stampaj_izraz(izrazi[i]);

    return 0;
}
```