Zadaci – 9. dio, *Funkcije (prvi dio, void-funkcije)*

Šta je funkcija?

Svaki C++ program ima bar jednu funkciju `main`. Kada vaš program počne s izvršavanjem, automatski se poziva `main`, koja može pozvati druge funkcije, od kojih će neke pozvati neke druge.

Dobro dizajnirane funkcije obavljaju specifičan i lahko razumljiv zadatak. Komplikovane zadatke bi trebalo rastaviti na više funkcija. Te funkcije se, u glavnom programu, koriste tako da se navede prototip funkcije (prije funkcije `main`) a potom se pozivaju iz funkcije `main`.

Postoje dva tipa funkcija: korisnički-definisane i ugrađene. Ugrađene funkcije su dio paketa vašeg kompajlera (biblioteke funkcija) – njih obezbjeđuje proizvođač, primjer: `sqrt(), pow(), sin(), cos()`…

Primjeri korisnički-definisanih funkcija

- Prilikom pokretanja nekog programa, prvo se izvršava funkcija `main`, bez obzira gdje se ona nalazila. U sljedećem primjeru izvršavat će se prvo linija br. 10.
- Ako funkcija `main` pozove neku drugu pomoćnu funkciju onda će se tek izvršiti pozvana funkcija.
- U liniji br. 14 se vrši poziv funkcije `ispisi_poruku`

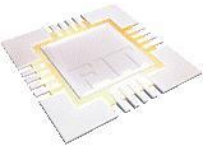
Analizirajte sljedeći program:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void ispisi_poruku()
5:   {
6:         cout << "a: Evo nas u funkciju 'ispisi_poruku' " << endl;
7:         cout << "b: Kraj - izlazak iz funkcije 'ispisi_poruku' " << endl;
8:   }
9:
10:  void main()
11:  {
12:        cout << "c: Evo nas u funkciji main" << endl;
13:
14:        ispisi_poruku();          //poziv funkcije
15:
16:        cout << "d: Evo nas OPET u funkciji main" << endl;
17:  }
```

Ovaj program se sastoji od :

- funkcije `main()`
- funkcije `ispisi_poruku()`

Kada se izvršava ovaj program, prvo se poziva glavna funkcija `main`, koja će ispisati poruku **c**, a zatim će se izvršiti naredba (u liniji br. 14) za poziv funkcije `ispisi_poruku`. Ova funkcija će ispisati poruku **a** i **b** (linija br. 6 i br. 7). Nakon završetka funkcije (izvršenja njene definicije), funkcija vraća vrijednost kroz svoje ime na mjesto odakle je pozvana, nastavlja se izvršavanje funkcije `main` od onog mjesta poziva funkcije (linija



br. 15), tj. izvršit će se naredba za ispis poruke **d** (linija br. 16). Zatim slijedi kraj čitavog programa zatvaranjem vitičaste zagrade funkcije `main` u liniji br. 17.

Znači, ovaj program će ispisat poruke sljedećim redoslijedom: c, a, b, d.

Iz ovog primjera se vidi da je ova najjednostavnija funkcija `ispisi_poruku` građena na isti način kao i funkcija `main`. Obje funkcije su tipa `void`, što znači da **nemaju** u tijelu naredbu **`return`**, i ne vraćaju nikakvu vrijednost onome ko ih poziva. Funkciju `main` poziva operativni sistem, a funkciju `ispisi_poruku` poziva funkcija `main`. Obje funkcije nemaju ulaznih parametara (argumenata), jer između zagrada – `main()` i `ispisi_poruku()` – nisu navedeni nikakvi parametri (varijable, vrijednosti).

Zadatak 89:

Koje će poruke ispisati sljedeći program? Zapišite redoslijed poruka i usporedite za rezultatom koji se nalazi na stranici br. 122, npr: b,d,e,c,e,d,a,d,…

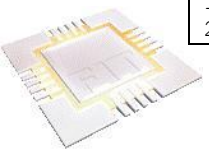
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void ispisi_poruku()
5:   {
6:         cout << "a: Evo nas u funkciju 'ispisi_poruku' " << endl;
7:         cout << "b: Kraj - izlazak iz funkcije 'ispisi_poruku' " << endl;
8:   }
9:
10:  void main()
11:  {
12:        ispisi_poruku();          //poziv funkcije
13:        cout << "c: Evo nas u funkciji main" << endl;
14:        ispisi_poruku();          //poziv funkcije
15:        ispisi_poruku();          //poziv funkcije
16:        cout << "d: Evo nas OPET u funkciji main" << endl;
17:        ispisi_poruku();          //poziv funkcije
18:  }
```

Rješenje se nalazi na kraju dokumenta.

Zadatak 90:

Koje će poruke ispisati sljedeći program? Zapišite redoslijed poruka i usporedite za rezultatom koji se nalazi na stranici br. 113, npr: b,d,e,c,e,d,a,d,…

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void ispisi_poruku1()
5:   {
6:         cout << "b: Evo nas u funkciju 'ispisi_poruku1' " << endl;
7:   }
8:
9:   void ispisi_poruku2()
10:  {
11:        cout << "d: Evo nas u funkciju 'ispisi_poruku2' " << endl;
12:  }
13:
14:  void main()
15:  {
16:        cout << "a: Evo nas u funkciji main" << endl;
17:
18:        ispisi_poruku1();
19:
20:        cout << "c: Evo nas OPET u funkciji main" << endl;
```



```
21:
22:         ispisi_poruku2();
23:         ispisi_poruku2();
24:         ispisi_poruku1();
25:
26:         cout << "e: Evo nas OPET u funkciji main" << endl;
27:
28:         ispisi_poruku1();
29:         ispisi_poruku2();
30:   }
```

Rješenje se nalazi na kraju dokumenta.

Opseg važenja varijabli

Obavezno zapamtite: Varijabla definirana u jednom bloku ne vrijedi (ne postoji) izvan tog bloka. (Blok čini bilo koji par vitičastih zagrada „**{ }**").

Opseg važenja varijable je samo blok u kojem smo je deklarisali. Slijede primjeri sa greškom:

Primjer br. 2:

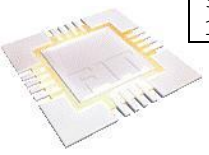
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:         if (10 > 1)
7:         {
8:               int a = 5;
9:         }
10:
11:        cout << "a = " << a << endl; // kompajler će prijaviti grešku:  'a' undeclared identifier
12:  }
```

Primjer br. 2:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:         if (10 > 1)
7:         {
8:               int b = 4;
9:         }
10:        else
11:        {
12:              int b = 5;
13:        }
14:        cout << "b = " << b << endl; //greska
15:  }
```

Primjer br. 3:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void main()
5:   {
6:         {
7:               int b = 4;
8:         }
9:         cout << "b = " << b << endl; //greska
10:  }
```



Isto tako i tijelo neke funkcije čini jedan blok. Pa sve lokalne varijable definirane u toj funkciji (bloku) ne vrijede u drugoj funkciji (bloku), evo primjera koji će prijaviti grešku:

Primjer br. 4:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void neka_funkcija()
5:   {
6:         cout << "a = " << a << endl; //Greška: 'a' : undeclared identifier
7:   }
8:
9:   void main()
10:  {
11:        int a;
12:        a = 12;
13:
14:        cout << "a = " << a << endl;
15:
16:        neka_funkcija();
17:  }
```

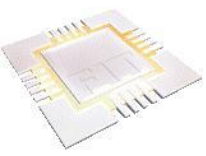
Opseg važenja globalnih varijabli

Globalne varijable vrijede u glavnoj funkciji `main` i u svim ostalim pomoćnim funkcijama datog programa, jer se deklaracija globalna varijable ne vrši ni u jednom bloku.

Globalne varijable se najčešće deklarišu na početku programa (npr. u liniji br. 3). Slijedi prepravljeni (prethodni) program sa globalnom varijablom `a` tipa *int:*

Primjer br. 4:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int a;  // ovo je sad globalna varijabla
5:
6:   void neka_funkcija()
7:   {
8:         cout << "a = " << a << endl;     // ispisat ce se poruka: 'a = 12'
9:   }
10:
11:  void main()
12:  {
13:        a = 12;
14:
15:        cout << "a = " << a << endl;     // ispisat ce se poruka: 'a = 12'
16:
17:        neka_funkcija();
     }
```



Zadatak 91:

Analizirajte sljedeći program. Koje će poruke ispisati?

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   int a = 10;  //globalna varijabla
5:
6:   void neka_funkcija()
7:   {
8:          a++;
9:          cout << "a = " << a << endl;
10:  }
11:
12:  void main()
13:  {
14:         a = 12;
15:         cout << "a = " << a << endl;
16:
17:         neka_funkcija();
18:
19:         cout << "a = " << a << endl;
20:
21:         neka_funkcija();
22:
23:         cout << "a = " << a << endl;
24:  }
```

Rješenje se nalazi na kraju dokumenta.

Zadatak 92:

Dovršite sljedeći program tako da radi na sljedeći način:

```
Koliko puta zelis da ti nesto kazem? ... 3
 1. Peace be upon you!
 2. Peace be upon you!
 3. Peace be upon you!
Kraj programa
```

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void neka_funkcija()
5:   {
6:          cout << "Peace be upon you! \n";
7:   }
8:
9:   void main()
10:  {
11:         int n;
12:         cout << "Koliko puta zelis da ti nesto kazem? ... ";
...          ...
22:  }
```

Rješenje se nalazi na kraju dokumenta.

Zadatak 93:

Napravite program u kome će te u funkciji `f1()` ispisati koliko ima brojeva (u opsegu od **1** do **10000**) koji su djeljivi sa 7 i koji će u funkciji `f2()` ispisati koliko ima brojeva koji su djeljivih sa 8 (u opsegu od **1** do **10000**).

U funkciji `main` se trebaju nalaziti samo sljedeće naredbe:

```
...   ...
32:   void main()
33:   {
34:           f1();          //poziv funkcije f1
35:           f2();          //poziv funkcije f2
36:   }
```

Rješenje se nalazi na kraju dokumenta.

Parametri funkcije

Do sada smo vidjeli da, kada pozivamo istu funkciji više puta, svaki put se ispisuje isti rezultat. Sada ćemo vidjeti kako za istu funkciju koju pozovemo možemo dobiti različite rezultate. To bi mogli iskoristiti u prethodnom programu (pogledajte program u rješenju zadatka br. 114).

Obje funkcije `f1()` i `f2()` su skoro iste, samo se razlikuju za broj 7, odnosno 8.

U funkciji `main` ćemo pozvati funkciju `f1`, ali sada ćemo proslijediti jedan *aktuelni parametar* toj funkciji prilikom poziva. **Aktuelni parametar** predstavlja vrijednost (broj ili varijabla) koja se nalazi u zagradama u pozivu funkcije:

```
...   ...
32:   void main()
33:   {
34:           f1(8);         //poziv funkcije f1 sa aktuelnim parametrom 8
35:           ...
36:   }
```

Ovdje smo u liniji br. 34 pozvali funkciju `f1` za vrijednost **8**. Poziv iste funkcije možemo vršiti i u liniji br. 35, ali sada sa vrijednošću **7**:

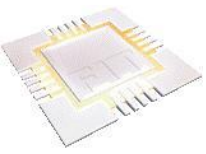
```
...   ...
32:   void main()
33:   {
34:           f1(8);         //poziv funkcije f1 za vrijednost 8
35:           f1(7);         //poziv funkcije f1 za vrijednost 7
36:   }
```

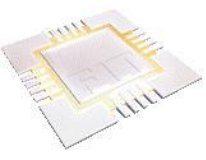
Kako možete vidjeti, funkcija `f2` nam je od ovog trenutka nepotrebna i možemo je izbrisati.

Funkciju `main` smo prepravili. Potrebno je još prepraviti i definiciju funkcije `f1`:

Možete primijeti da je parametar funkcije `f1` vrijednost tipa *integer* (broj 8 i broj 7).

Da bi funkcija `f1` znala šta da radi sa primljenim parametrom, moramo i to 'reći', tako što ćemo u zagradama (linija br. 4) dodati deklaraciju jedne varijable. Ime te varijable može biti bilo koji valjani identifikator. U rješenjima ćemo koristiti imena `u1`, `u2`, … što asocira na skraćenicu od riječi 'ulaz'. Ovakva varijabla koja je deklarisana u zaglavlju (definicije) funkcije kao ulazni parametar se naziva **formalni parametar**.





Zadatak 94:

Prepravite funkciju `f1`. Dodajte jedan formalni parametar `u1` tipa *int*.

Rješenje se nalazi na kraju dokumenta.

Pomoć:

Sada ćemo umjesto konstante 8, korištene u funkciji `f1`, koristiti varijablu `u1`. Varijabla `u1` će, kada se funkcija prvi put pozove, imati vrijednost 8, dok će drugi put imati vrijednost 7.

Funkcija sa više parametara

Umjesto običnih brojeva (7, 8) možete prosljeđivati i varijable. A možete prosljeđivati i više parametara u funkciju. Oni se, u pozivu funkcije, moraju navesti istim redoslijedom kao u prototipu i zaglavlju definicije funkcije. Evo primjera poziva funkcije `f1` koja sada ima tri aktuelna parametra, a to su:

1. broj sa kojim se vrši provjera djeljivosti
2. početak niza
3. kraj niza

```
...   ...
18:   void main()
19:   {
20:         f1(8, 1, 1000);     //poziv funkcije f1
21:         f1(7, 1, 1000);     //poziv funkcije f1
22:   }
```

Pošto imamo tri aktuelna parametra, moramo deklarisati tri formalna parametra:

```
...   ...
4:    void f1(int u1, int u2, int u3)
5:    {
6:    ...
```

Sada možemo u funkciji `f1` umjesto brojeva 1 i 10000 koristiti varijable `u2` i `u3`.

Zadatak 95:

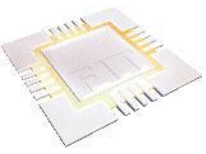
Prepraviti prethodni program tako da u glavnoj funkciji `main` zahtijevate od korisnika da unese dva cijela broja - `m` za početak i `n` za kraj niza, zatim još dva cijela broja - `a` i `b`. Program treba da pomoću funkcije `f1`:

- ispiše sa koliko brojeva iz niza [`m`, `n`] je djeljiv broj `a`
- ispiše sa koliko brojeva iz niza [`m`, `n`] je djeljiv broj `b`

Rješenje se nalazi na kraju dokumenta.

Pomoć:

- u funkciji `main` ćemo izvršiti unos za `m`, `n`, `a`, `b`
- pozvat ćemo funkciju `f1` i proslijediti `a` (umjesto 8), kao i vrijednosti `m` i `n`



- pozvat ćemo funkciju `f1` i proslijediti `b` (umjesto 7), kao i vrijednosti `m` i `n`
- u funkciji `f1` ćemo primiti parametre kao `u1`, `u2` i `u3` koji su tipa *int*
- kao u prethodnom program, izvršiti ćemo *for*-petlju, ali sada ćemo koristiti varijable `u2` i `u3`, umjesto 1 i 10000
- u funkciji ćemo ispisati vrijednost varijable `brojac`
- funkciji `f1` ćemo proslijediti akuelne parametre redoslijedom: (`a`, `m`, `n`), odnosno (`b`, `m`, `n`), a primat ćemo ih redoslijedom `u1`, `u2`, `u3`, što znači da će u funkciji `f1`:
  - `u1` preuzeti vrijednost `a`, odnosno `b`
  - `u2` preuzeti vrijednost `m`
  - `u3` preuzeti vrijednost `n`

Zadatak 96:

Napravite program u kome će te, pomoću jedne funkcije, ispisati sumu *parnih brojeva* i sumu *kvadrata neparnih* brojeva od `m` do `n`. U funkciji `main` ćete od korisnika tražiti da unese cijeli broj `m` za početak niza i `n` za kraj niza.

Rješenje se nalazi na kraju dokumenta.

Pomoć:

- u funkciji `main` ćemo izvršiti unos za `m`, `n`
- pozvat ćemo funkciju `f1` i proslijediti, kao aktuelne parametre, vrijednosti `m` i `n`
  - u funkciji `f1` ćemo primiti parametre kao `u1`, `u2` koji su tipa *integer*
  - postaviti ćemo varijable `suma1` i `suma2` na nulu koje su tipa *int*,
  - `suma1` će zbrajati sve parne brojeve
  - `suma2` će zbrajati sve neparne brojeve
  - izvršiti ćemo *for*-petlju sa brojačem `i` koji se mijenja od `u1` do `u2`
    - provjeriti ćemo da li je broj `i` paran, tj. da li je `i` djeljivo sa 2:

      ako jeste: onda ćemo povećati `suma1` za `i`

      ako nije: onda ćemo povećati `suma2` za `i*i`
  - ispisati ćemo `suma1` i `suma2` na ekran

Zadatak 97:

Napravite program koji će u funkciji `main` zahtijevati od korisnika da unese dimenzije bazena (širina, dužina, visina) u metrima i koji će pomoću funkcije izračunati i ispisati zapreminu bazenu u litrama. Deklarišite prototip funkcije.

Rješenje se nalazi na kraju dokumenta.

Prototip (deklaracija) funkcije

Funkcije u programi treba koristiti na takav način da se prvo (u polju deklaracija – prije funkcije `main`) navede njena deklaracija (prototip) a da se definiše van funkcije `main`. Deklaracija „saopštava" kompajleru povratni tip, ime i parametre funkcije i tipove parametara. Ni jedna funkcija se ne može pozvati iz neke druge funkcije ako nije prvo deklarisana. Deklaracija se naziva ***prototip***.

Postoje dva načina da se deklariše funkcija:

1. Zapisivanjem prototipa u istu datoteku u kojoj se nalazi vaša glavna funkcija `main`, tj. iznad funkcije `main`.
2. Definisanjem funkcije (prije funkcije `main` ili u funkciji `main`) prije nego što je pozove neka druga funkcija. U ovom slučaju nije potrebno navoditi deklaraciju funkcije.

Preporučuje se korištenje funkcija na način br. 1.

U svim prethodnim programima nije korišten prototip funkcije.

Prototip je sličan zaglavlju definicije funkcije, tj. sličan je definiciji funkcije bez tijela, samo što završava na tačka-zarez (**;**). Evo razlike:

Preporučuje se ovaj

bez prototipa – slučaj 2

```
#include <iostream>
using namespace std;

void volumen(int u1, int u2, int u3)
{
    long int V;

    V = u1 * u2 * u3 * 1000;

    cout << "Zapremina: " << V << " m \n";
}

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

    cout << "Unesi dimenzije u metrima:\n";
    cin >> a >> b >> c;

    volumen(a,b,c);
}
```

sa prototipom - slučaj 1

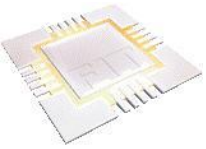
```
#include <iostream>
using namespace std;

void volumen(int, int, int);

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

    cout << "Unesi dimenzije u metrima:\n";
    cin >> a >> b >> c;

    volumen(a,b,c);
}

void volumen(int u1, int u2, int u3)
{
    long int V;

    V = u1 * u2 * u3 * 1000;

    cout << "Zapremina: " << V << " m \n";
}
```

U prototipu se ne moraju navoditi imena varijabli (formalnih parametara). Ako ih koristimo, možemo koristiti imena parametara koja se razlikuju od imena parametara u definiciji funkcije. U prethodnom primjeru smo mogli i ovako navesti prototip:

```
void volumen(int BilaSta, int OvoNijePotrebno, int OviNaziviSeIgnorisu);
```

Preporuka je ne navoditi nikakva imena varijabli u prototipu, jer prototip služi samo da „kaže" kompajleru tip funkcije (u našem slučaju je to `void`), ime funkcije i tip ulaznih argumenata. U definiciji funkcije moramo tačno navoditi ona imena varijabli koja ćemo koristiti u tijelu definicije funkcije (u našem slučaju: **u1**, **u2**, **u3**). U daljnjim zadacima ćemo koristiti i prototip, ali bez imena varijabli:

```
void volumen(int, int, int);
```



Zadatak 98:

Napravite program koji će od korisnika zahtijevati unos dva broja, `m` za početak niza i `n` za kraj niza. Funkcija `main` treba pozvati funkciju `f1` za svaki cijeli broj iz tog niza [`m`, `n`]. Funkcija `f1` treba provjeriti da li je broj koji ona prima kvadrat nekog broja, tj. da li korijen tog broja cijeli broj. Samo ako jeste, funkcija treba broj ispisati na ekran.

Rješenje se nalazi na kraju dokumenta.

Pomoć:

- u funkciji `f1` treba provjeriti da li je `u1` pozitivan broj
- da bi provjerili da li je korijen broja `u1` cijeli broj moramo učiniti sljedeće:
- vrijednost korijena od `u1` ćemo smjestiti u varijablu `korijen_f` koja je tipa ***float***
- vrijednost korijena od `u1` ćemo smjestiti u varijablu `korijen_i` koja je tipa ***int***
- pošto je varijabla `korijen_i` tipa *int*, decimalni dio (ako postoji) će se zanemariti
- ako je `korijen_f` jednako `korijen_i`, što znači je vrijednost `korijen_f` cijeli broj, ispisat ćemo na ekran broj `u1`

Rješenja

Rješenje zadatka br. **89**:
a, b, c, a, b, a, b, d, a, b

Rješenje zadatka br. **90**:
a, b, c, d, d, b, e, b, d

Rješenje zadatka br. **91**:

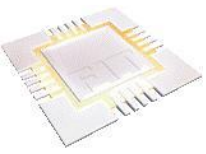
Opis programa:

- kada se pokrene program varijabla **a** će imati vrijednost 10, ali se neće nigdje ispisati
- poziva se funkcija `main()`:
  - varijabla **a** dobiva vrijednost 12 `// linija broj 14`
  - "**a=12**" se ispisuje na ekran `// linija broj 15`
  - iz `main` poziva se `neka_funkcija()` `// linija broj 17`
    - **a** se inkrementira sa 12 na 13 `// linija broj 8`
    - "**a=13**" se ispisuje na ekran `// linija broj 9`
  - "**a=13**" se ispisuje na ekran `// linija broj 19`
  - iz `main` poziva se `neka_funkcija()` `// linija broj 21`
    - **a** se inkrementra sa 13 na 14 `// linija broj 8`
    - "**a=14**" se ispisuje na ekran `// linija broj 9`
  - "**a=14**" se ispisuje na ekran `// linija broj 23`

Korištenje globalnih varijabli nije preporučeno, ali je moguće. Za korištenje vrijednosti varijabli u različitim funkcijama koristit ćemo prosljeđivanje više varijabli kao parametre (argumente) funkcijama pri njihovom pozivu, kao i vraćanje jedne izlazne vrijednosti kroz `return` za funkcije koje nisu tipa `void`, a o tome malo kasnije.

Nastavljamo ali obavezno bez globalnih varijabli.

Rješenje zadatka br. **92**:



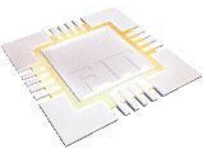
```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void neka_funkcija()
5:   {
6:         cout << "Peace be upon you! \n";
7:   }
8:
9:   void main()
10:  {
11:        int n;
12:        cout << "Koliko puta zelis da ti nesto kazem? ... ";
13:        cin >> n;
14:
15:        for (int i=1; i<=n; i++)
16:        {
17:              cout << " " << i << ". ";
18:              neka_funkcija();
19:        }
20:
21:        cout << "Kraj programa \n";
22:  }
```

Rješenje zadatka br. **93**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void f1()
5:   {
6:         int brojac;         //varijabla koja vrijedi samo za f1()
7:         brojac = 0;
8:
9:         for (int i=1; i<=10000; i++)
10:        {
11:              if (i%8 == 0)
12:                    brojac++;
13:        }
14:
15:        cout << "Ukupno djeljivih brojeva sa 8 ima " << brojac << endl;
16:  }
17:
18:  void f2()
19:  {
20:        int brojac;         //varijabla koja vrijedi samo za f2()
21:        brojac = 0;
22:
23:        for (int i=1; i<=10000; i++)
24:        {
25:              if (i%7 == 0)
26:                    brojac++;
27:        }
28:
29:        cout << "Ukupno djeljivih brojeva sa 7 ima " << brojac << endl;
30:  }
31:
32:  void main()
33:  {
34:        f1();         //poziv funkcije f1
35:        f2();         //poziv funkcije f2
36:  }
```

U obje funkcije `f1()` i `f2()` smo mogli koristili isti naziv za varijablu `brojac`. To su različite varijable koje imaju isto ime a kao da se nalaze u različitom prostoru, jer je njihov opseg važenja samo u onoj funkciji (bloku) u kojoj su i deklarisane.

Rješenje zadatka br. **94**:

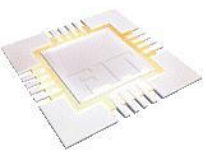


```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void f1(int u1)     //  Svi parametri proslijeđeni funkciji f1 primaju se kao integer u varijablu u1.
5:   {                                                    //  Varijabla u1 vrijedi samo za funkciju f1.
6:          int brojac;
7:          brojac = 0;
8:
9:          for (int i=1; i<=10000; i++)
10:         {
11:                if (i%u1 == 0)
12:                       brojac++;
13:         }
14:
15:         cout << "Ukupno djeljivih brojeva sa " << u1 << " ima " << brojac << endl;
16:  }
17:
18:  void main()
19:  {
20:         f1(8);        //poziv funkcije f1
21:         f1(7);        //poziv funkcije f1
22:  }
```

Rješenje zadatka br. **95**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void f1(int u1, int u2, int u3)
5:   {
6:          int brojac;
7:          brojac = 0;
8:
9:          for (int i=u2; i<=u3; i++)
10:         {
11:                if (i%u1 == 0)
12:                       brojac++;
13:         }
14:
15:         cout << "Ukupno djeljivih brojeva sa " << u1 << " ima " << brojac << endl;
16:  }
17:
18:  void main()
19:  {
20:         int m, n, a, b;
21:
22:         cout << "Unesi prvi i zadnji broj niza: \n";
23:         cin >> m >> n;
24:         cout << "Unesi jos dva broja: \n";
25:         cin >> a >> b;
26:
27:         f1(a, m, n);     // poziv funkcije f1 za a, m, n
28:         f1(b, m, n);     // poziv funkcije f1 za b, m, n
29:  }
```

Rješenje zadatka br. **96**:



```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void f1(int u1, int u2)
5:   {
6:          long int suma1, suma2;
7:
8:          suma1 = 0;
9:          suma2 = 0;
10:
11:         for (int i=u1; i<=u2; i++)
12:         {
13:                if (i%2 == 0)
14:                       suma1 = suma1 + i;
15:                else
16:                       suma2 = suma2 + i*i;
17:         }
18:
19:         cout << "za niz[" << u1 << ", " << u2 << "]" << endl;
20:         cout << "suma parnih brojeva: " << suma1 << endl;
21:         cout << "suma kvadrata neparnih brojeva: " << suma2 << endl;
22:  }
23:
24:  void main()
25:  {
26:         int m, n;
27:         cout << "Unesi pocetak i kraj niza \n";
28:         cin >> m >> n;
29:
30:         f1(m, n);     //poziv funkcije f1 za m, n
31:  }
```

Rješenje zadatka br. **97**:

```
1:   #include <iostream>
2:   using namespace std;
3:
4:   void volumen(int u1, int u2, int u3)
5:   {
6:         long int V;
7:
8:         V = u1 * u2 * u3 * 1000;     // kubni metar pomnozen sa 1000 daje litar
9:
10:        cout << "Zapremina: " << V << " m \n";
11:  }
12:
13:  void main()
14:  {
15:        int a, b, c;
16:
17:        cout << "Unesi dimenzije u metrima:\n";
18:        cin >> a >> b >> c;
19:
20:        volumen(a,b,c);
21:  }
```

Rješenje zadatka br. **98**:

```
1:   #include <iostream>
2:   #include <math.h>
3:   using namespace std;
4:
5:   void f1(int); // prototip (deklaracija) funkcije
6:
7:   void main()
8:   {
9:         int m, n;
10:
11:        cout << "Unesi pocetak i kraj niza: \n";
12:        cin >> m >> n;
13:
14:        for (int i=m; i<=n; i++)
15:        {
16:              f1(i);
17:        }
18:  }
19:
20:  void f1( int u1)
21:  {
22:        if (u1>=0)
23:        {
24:              float korijen_f;
25:              korijen_f = sqrt(float(u1));
26:
27:              int korijen_i;
28:              korijen_i = korijen_f;    // korijen_i će zanemariti decimalni dio od korijen_f
29:
30:              if (korijen_i == korijen_f)
31:                  cout << u1 << " = " << korijen_i << " * " << korijen_i << endl;
32:        }
33:  }
```