Programiranje 1

Programski jezik C

— Zadaci sa vežbi —

Milena Vujošević - Janičić 2011

Sadržaj

Predgovor

Ovo je prateći materijal za vežbe koje držim iz predmeta Programiranje 1. On ne može zameniti pohađanje vežbi niti korišćenje druge preporučene literature.

Veliki deo materijala čine zadaci i rešenja mr Filipa Marića (raspoloživi na www.matf.bg.ac.rs/~filip/pp/0405/index.pl). Takođe korišćen je i materijal sa sajta koleginice Jelene Grmuše www.matf.bg.ac.rs/~jelenagr i kolege Miroslava Marića www.matf.bg.ac.rs/~maricm. Tekstovi i objašnjenja su uglavnom zasnovani na knjizi *Programski jezik C*, autora *Kerninghan & Ritchie*.

Zahvaljujem svojim studentima na aktivnom učešću u nastavi čime su mi pomogli u uobličavanju ovog materijala.

Svi komentari i sugestije vezane za ovaj materijal biće veoma dobrodošli.

Milena Vujošević-Janičić
www.matf.bg.ac.rs/~milena

1

Programski jezik C

1.1 Zdravo svete!

Primer 1.1.1 *Program štampa poruku "hello, world".*

```
#include <stdio.h>

main()
/*iskazi f-je main su zatvoreni u zagrade */
{
/*poziv f-je printf da odstampa poruku*/
printf("hello, world\n");
}
```

Primer 1.1.2 *Program štampa poruku "hello, world"*

```
#include <stdio.h>

main()
{
printf("hello, ");
printf("world");
printf("\n");
}
```

Specijalni znaci:

```
\n novi red
\t tabulator
\\ kosa crta
\" navodnici
\a zvuk
\' jednstruki navodnik
```

1.2 Imena promenljivih

Postoje ograničenja: u imenu se mogu pojaviti slova i cifre, potcrta "_" se smatra slovom.
Velika i mala slova se razlikuju.
`int x, X; /*To su dve razlicite promenljive!!!*/`
Ključne reči kao što su if, else, for, while, se ne mogu koristiti za imena promenljivih.

1.3 Deklaracije

Da bi se promenljiva mogla upotrebljavati ona se mora na početku programa deklarisati. Prilikom deklaracije može se izvršiti i početna inicijalizacija.

```
int broj; /*Deklaracija celog broja*/
int vrednost=5; /*Deklaracija i inicijalizacija celog broja*/
```

Kvalifikator const može biti dodeljen deklaraciji bilo koje promenljive da bi označio da se ona neće menjati

```
const double e=2.71828182845905
```

1.4 Tipovi i veličina podataka

Osnovni tipovi podataka:

```
int     ceo broj
char    znak, jedan bajt
float   realan broj
double  realan broj dvostruke tacnosti
```

```
char    jedan bajt, sadrzi jedan znak
int     celobrojna vrednost, 2 ili 4 bajta
float   realan broj, jednostruka tacnost
double dvostruka tacnost
```

Postoje kvalifikatori koje pridružujemo osnovnim tipovima short(16) i long(32):

```
short int kratak_broj;
long int dugacak_broj;
short kratak;
long dugacak;
```

Važi

$$broj\_bajtova(short) <= broj\_bajtova(int) <= broj\_bajtova(long)$$

Tip char zauzima jedan bajt ali u zavisnosti od sistema ovaj tip može da se odnosi na označene ili na neoznačene brojeve. Zato Postoje kvalifikatori `signed` i `unsigned` koji preciziraju da li se misli na označene ili neoznačene cele brojeve. Npr.
`signed char`: -128 do 127
dok je
`unsigned char`: od 0 do 255.

Veličina za `int` je različita u zavisnosti od sistema i može biti 2 ili 4 bajta. Međutim, ako je promenljiva tipa `short int` onda ona sigurno zauzima samo dva bajta a to znači da u nju mogu da stanu celobrojne vrednosti iz intervala -32 768 do +32 767, odnosno mogu se koristiti sledeći sinonimi

```
short <=> short int <=> signed short int <=> -32 768 do 32 767
```

Ukoliko se koristi `unsigned short int` onda je interval
`unsigned short int <=> 0 do 65 535`

Za realne tipove podataka koriste se `float`, `double` i `long double`. Njihove veličine mogu da zavise od sistema. Na primer:

```
float (4 bajta)
float minimalna pozitivna vrednost 1.175494351e-38
float maximalna pozitivna vrednost 3.402823466e+38

double (8 bajta)
double minimalna pozitivna vrednost 2.2250738585072014e-308
double maximalna pozitivna vrednost 1.7976931348623158e+308

long double (10 bajta)
long double minimalna pozitivna 3.362103143112093506 3e-4932
long double maximalna pozitivna 1.189731495357231765e+4932
```

Primer 1.4.1 *Uvođenje promenljivih u program.*

```
#include <stdio.h>

main()
 {
 /*deklaracija vise promenljivih
 istog tipa */
 int rez,pom1,pom2;
 pom1=20;
```

```
pom2=15;
rez=pom1-pom2;

/*ispisivanje rezultata*/
printf("Rezultat je %d-%d=%d\n",pom1,pom2,rez);
}
```

Izlaz iz programa:
Rezultat je 20-15=5

Iskaz dodele:
pom1=20;
pom2=15;
Individualni iskazi se zavrsavaju sa ;

1.5 Konstante

Koji su tipovi konstanti?

Celobrojna konstanta 1234 je tipa int.
Da bi konstanta bila long navodi se iza nje slovo L ili l, npr 123456789L.
Ako želimo da nam je konstanta unsigned onda na kraju pišemo U ili u.
Može i 1234567ul.

Konstante realnih brojeva sadrže decimalnu tačku(123.4) ili eksponent(1e-2) ili i jedno i drugo. Njihov tip je double osim ako nemaj sufiks f ili F kada je u pitanju float. L ili l označavaju long double.

Oktalna konstanta počinje sa 0, a heksadecimalna sa 0x. Npr broj 31 ili 037 - oktalno ili 0x1f - heksadecimalno. I one mogu da imaju U i L na kraju.

Znakovna konstanta je celobrojna vrednost napisana između jednostrukih navodnika. Vrednost date konstante je numerička vrednost datog znaka u računarskom setu znakova. Npr možemo da pišemo '0' umesto 48.
!!!Razlikovati znakovne konstante i niske koje se navode između dvostrukih navodnika!

Posebni znaci su znak za kraj reda `'\n'`, tab `'\t'` i slično. Iako se zapisuju kao više znakova, oni označavaju samo jedan znak i njima takođe odgovara samo jedan bajt.

Znakovna konstanta `'\0'` predstavlja znak čija je vrednost nula, treba ga razlikovati od '0' koja je znak čija je vrednost 48.

Primer 1.5.1 *Koja je vrednost konstantnog izraza*

1. `0x10 + 020 + '9' - '1'`

2. `0x20 + 010 + '8' - '0'`

1.6 Funkcije printf i scanf

```
printf("%d\t%d\n", broj1, broj2);
uvek je prvi argument izmedju " "
%d ceo broj
\t tab izmedju
\n novi red
Svaka % konstrukcija je u paru sa argumentom koji sledi.
```

Primer 1.6.1

```
#include <stdio.h>
main()
  {
  printf("Slova:\n%3c\n%5c\n", 'z' , 'Z');
  }
```

Izlaz iz programa:

```
Slova:
  z
    Z
```

`%c` *je za stampanje karaktera*
`%3c` *je za stampanje karaktera na tri pozicije*
Isto tako smo mogli i `%3d` *za stampanje broja na tri pozicije ili* `%6d` *za stampanje broja na 6 pozicija.*

Pravila:
`%d` stampaj kao ceo broj
`%6d` stampaj kao ceo broj sirok najvise 6 znakova
`%f` stampaj kao realan broj
`%6f` stampaj kao realan broj sirok najvise 6 znakova
`%.2f` stampaj kao realan broj sa dve decimale
`%6.2f` stampaj kao realan broj sirok najvise 6 znakova a od toga 2 iza decimalne tacke
`%c` karakter
`%s` string
`%x` heksadecimalni broj
`%%` je procenat

Primer 1.6.2 *Ispisivanje karaktera:* `%c`*, ispisivanje asci vrednosti karaktera* `%d`

```
#include <stdio.h>
main()
{
int vrednost;
vrednost='A';
printf("Veliko slovo\n karakter=%3c\nvrednost=%3d\n",vrednost,vrednost);
vrednost='a';
printf("Malo\n karakter=%3c\nvrednost=%3d\n",vrednost,vrednost);
}

Izlaz (u slucaju ASCII):
Veliko slovo
karakter= A
vrednost= 65
Malo
karakter= a
vrednost= 97
```

Primer 1.6.3 *Prikazuje unos celog broja koristeci* `scanf("%d", &x)`

```
#include <stdio.h>

main()
{
    int x;
    printf("Unesi ceo broj : ");

    /* Obratiti paznju na znak &
       (operator uzimanja adrese)
       pre imena promenljive u funkciji
       scanf */
    scanf("%d",&x);

    /* U funkciji printf nije
    potrebno stavljati &  */
    printf("Uneli ste broj %d\n", x);
}
```

Primer 1.6.4 *Program sabira dva uneta cela broja*

```
#include <stdio.h>

main()
{
```

```
    int a, b, c;
    printf("Unesi prvi broj : ");
    scanf("%d", &a);
    printf("Unesi drugi broj : ");
    scanf("%d", &b);
    c = a + b;
    printf("%d + %d = %d\n", a, b, c);
}
Ulaz:
Unesi prvi broj : 2 <enter>
Unesi drugi broj : 3 <enter>
Izlaz:
2 + 3 = 5
```

1.7 Aritmetički operatori

```
+ - * /
% (samo za celobrojne vrednosti)
unarno + i -
```

Asocijativnost sleva na desno, prioritet kao u matematici.

Primer 1.7.1 *Program ilustruje neke od aritmetičkih operacija.*

```
#include <stdio.h>
main()
{
int a, b;
printf("Unesi prvi broj : ");
scanf("%d",&a);

printf("Unesi drugi broj : ");
scanf("%d",&b);

/* Kada se saberu dva cela broja, rezultat je ceo broj*/
printf("Zbir a+b je : %d\n",a+b);
/* Kada se oduzmu dva cela broja, rezultat je ceo broj*/
printf("Razlika a-b je : %d\n",a-b);
/* Kada se pomnoze dva cela broja, rezultat je ceo broj*/
printf("Proizvod a*b je : %d\n",a*b);
/* Kada se podele dva cela broja, rezultat je ceo broj!!!*/
printf("Celobrojni kolicnik a/b je : %d\n", a/b);
/* Rezultat je ceo broj, bez obzira sto ga ispisujemo kao realan*/
```

```
printf("Pogresan pokusaj racunanja realnog kolicnika a/b je : %f\n", a/b);
/* Eksplicitna konverzija, a i b pretvaramo u relane brojeve kako
   bi deljenje bilo realno*/
printf("Realni kolicnik a/b je : %f\n", (float)a/(float)b);
/* Ostatak pri deljenju se moze izvrsiti samo nad celim brojevima*/
printf("Ostatak pri deljenju a/b je : %d\n", a%b);
}

Ulaz:
Unesi prvi broj : 2 <enter>
Unesi drugi broj : 3 <enter>
Izlaz:
Zbir a+b je : 5
Razlika a-b je : -1
Proizvod a*b je : 6
Celobrojni kolicnik a/b je : 0
Progresan pokusaj racunanja realnog kolicnika a/b je : 0.000000
Realni kolicnik a/b je : 0.666667
Ostatak pri deljenju a/b je : 2
```

Primer 1.7.2 *Program ilustruje celobrojno i realno deljenje.*

```
#include <stdio.h>

main()
{
int a = 5;
int b = 2;
int d = 5/2;    /* Celobrojno deljenje - rezultat je 2 */
float c = a/b;  /* Iako je c float, vrsi se celobrojno
                        deljenje jer su i a i b celi */

/* Neocekivani rezultat 2.000000 */
printf("c = %f\n",c);
printf("Uzrok problema : 5/2 = %f\n", 5/2);
printf("Popravljeno : 5.0/2.0 = %f\n", 5.0/2.0);
printf("Moze i : 5/2.0 = %f i 5.0/2 = %f \n", 5/2.0, 5.0/2);
printf("Za promenljive mora kastovanje : %f\n", (float)a/(float)b);

}

Izlaz iz programa:
c = 2.000000
Uzrok problema : 5/2 = 2.000000
```

```
Popravljeno : 5.0/2.0 = 2.500000
Moze i : 5/2.0 = 2.500000 i 5.0/2 = 2.500000
Za promenljive mora kastovanje : 2.500000
```

Zadatak 1 *Šta će biti ispisano nakon izvršavanja sledećeg programa?*

```
#include <stdio.h>
main()
 {
     int x=506, y=3, z=21, t=2;
     printf("x=%d y=%d\n",x,y);
     printf("z - t=%d\n", z-t);
     printf("z / t =%d\n",z / t);
     printf("-x=%d\n",- x);
     printf("x %% y=%d\n", x%y);
}
```

1.8 Konverzija

1.8.1 Automatska konverzija

Ako je jedan od operanada različličit vrši se konverzija, uvek u smeru manjeg ka većem tipu

Naredba dodele:

```
int i=5;
float f=2.3;
f=i; /* f ce imati vrednost 5.0*/

obrnuto:

int i=5;
float f=2.3;
i=f; /* i ce imati vrednost 2*/
```

1.8.2 Eksplicitna konverzija

```
(tip)<izraz>

float x;
x=2.3+4.2;          /* x ce imati vrednost 6.5 */
x=(int)2.3+(int)4.2; /* x ce imati vrednost 6 */
x=(int)2.3*4.5;     /* x ce imati vrednost 9.0 jer zbog prioriteta
```

```
                    operatora konverzije prvo ce biti izvrsena
                    konverzija broja 2.3 u 2 pa tek onda izvrseno
                    mnozenje. */
x=(int)(2.3*4.5)    /* x ce imati vrednost 10.0 */
```

Primer 1.8.1 *Kako izbeći celobrojno deljenje*

```
int a,b;
float f;
a = 5;
b = 2;
f = a/b; /* Celobrojno deljenje, f=2*/
f = (1.0*a)/b;  /* Implicitna konverzija: 1.0*a je realan
                broj pa priliko deljenja sa b dobija se
                realan rezultat f=2.5*/
f = (0.0+a)/b;  /* Implicitna konverzija: (0.0+a) je realan
                broj pa priliko deljenja sa b dobija se
                realan rezultat f=2.5*/
f = (float)a/(float)b; /* Eksplicitna konverzija*/
```

1.9 Operator `sizeof()`

Primer 1.9.1 *Demonstracija sizeof operatora.* ***`Sizeof`*** *operator izračunava veličinu tipa odnosno promenjive.*

```
#include<stdio.h>
main()
{
int i;
float f;

printf("sizeof(int)=%d\n", sizeof(int));
printf("sizeof(long)=%d\n", sizeof(long));
printf("sizeof(short)=%d\n", sizeof(short));
printf("sizeof(signed)=%d\n", sizeof(signed));
printf("sizeof(unsigned)=%d\n", sizeof(unsigned));
printf("sizeof(char)=%d\n", sizeof(char));
printf("sizeof(float)=%d\n", sizeof(float));
printf("sizeof(double)=%d\n", sizeof(double));

printf("sizeof(i)=%d\n", sizeof(i));
printf("sizeof(f)=%d\n", sizeof(f));
}
```

```
Izlaz iz programa (u konkretnom slucaju):
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
sizeof(signed)=4
sizeof(unsigned)=4
sizeof(char)=1
sizeof(float)=4
sizeof(double)=8
sizeof(i)=4
sizeof(f)=4
```

1.10 Operatori i izrazi dodeljivanja vrednosti

```
i = i + 2;
ekvivalento je sa
i+=2;

Moze i za:
+ - * / % << >> ^ | && ||
izraz1 op = izraz2
je ekvivalnetno sa
izraz1 = (izraz1) op (izraz2)

x*= y+1 je ekvivalento sa x = x * (y+1)
```

Takvo pisanje je krace i efikasnije.

1.11 Inkrementacija i dekrementacija

Operatori ++ i --

```
x=++n; se razlikuje od x=n++;

y=(x++)*(++z);
```

Primer 1.11.1 *Ilustracija prefiksnog i postfiksnog operatora ++*

```
#include <stdio.h>
main()
{
int x, y;
int a = 0, b = 0;
```

```
printf("Na pocetku : \na = %d\nb = %d\n", a, b);

/* Ukoliko se vrednost izraza ne koristi, prefiksni i
    postfiksni operator se ne razlikuju */
a++;
++b;
printf("Posle : a++; ++b; \na = %d\nb = %d\n", a, b);

/* Prefiksni operator uvecava promenljivu, i rezultat
    je uvecana vrednost */
x = ++a;

/* Postfiksni operator uvecava promenljivu, i rezultat je
stara (neuvecana) vrednost */
y = b++;

printf("Posle : x = ++a; \na = %d\nx = %d\n", a, x);
printf("Posle : y = b++; \nb = %d\ny = %d\n", b, y);
}

Izlaz iz programa:
Na pocetku:
a = 0
b = 0
Posle : a++; ++b;
a = 1
b = 1
Posle : x = ++a;
a = 2
x = 2
Posle : y = b++;
b = 2
y = 1
```

1.12 Relacioni i logički operatori

Relacioni operatori:

```
>   >=      <   <= isti prioritet
== !=               nizi prioritet

(3<5)
```

```
(a<=10)
 a < 5 != 1  <=> (a < 5)!=1
```

Logicki operatori:

```
! unarna negacija (najvisi prioritet)
&& logicko i (visi prioritet od ili)
|| logicko ili izracunavaju se sleva na desno!

5 && 4 vrednost je tacno
10 || 0 vrednost je tacno
0 && 5 vrednost je 0
!1 vrednost je 0
!9 vrednost je 0
!0 vrednost je 1
!(2>3) je 1
a>b && b>c || b>d je isto sto i ((a>b) && (b>c)) || (b>d)
koja je vrednost ako je a=10, b=5, c=1, d=15?
```

Primer 1.12.1 *Ilustracija logičkih i relacijskih operatora.*

```
#include <stdio.h>

main()
{
    int a = 3>5,  /* manje */
    b = 5>3,  /* vece */
    c = 3==5, /* jednako */
    d = 3!=5; /* razlicito */

    printf("3>5 - %d\n5>3 - %d\n3==5 - %d\n3!=5 - %d\n", a, b, c, d);

    /*Lenjo izracunavanje: kako 3 nije vece od 5 to se vrednost
    drugog poredjenja nece racunati jer je netacno u konjunkciji
    sa proizvoljnim izrazom sigurno netacno. */
    printf("Konjunkcija : 3>5 && 5>3 - %d\n", a && b);

    /*Lenjo izravunavanje: tacno u disjunkciji sa proizvoljnim
    izrazom daje tacno tako da se vrednost izraza 3>5 nece
    izracunavati*/
    printf("Disjunkcija : 5>3 || 3>5 - %d\n", b || a);
    printf("Negacija : !(3>5) - %d\n", !a);

}
```

```
Izlaz iz programa:
3>5 - 0
5>3 - 1
3==5 - 0
3!=5 - 1
Konjunkcija : 3>5 && 5>3 - 0
Disjunkcija : 3>5 || 5>3 - 1
Negacija : !(3>5) - 1
```

Primer 1.12.2 *Prilikom izracunavanja izraza A&&B, ukoliko je A netačno, izraz B se ne izračunava. Prilikom izračunavanja izraza A || B, ukoliko je A tačno, izraz B se ne izračunava.*

```
#include <stdio.h>
main()
{
    int a = 3, b = 4, c = 5, d = 6;

    if((a>b)&&(++c<d)) a++;
    /*a = 3, b = 4, c = 5, d = 6*/
    printf("a = %d\n b=%d\n c=%d\n d=%d\n");

    if((a<b)||(c<d++)) b++;
    /*a = 3, b = 5, c = 5, d = 6*/
    printf("a = %d\n b=%d\n c=%d\n d=%d\n");
}
```

1.13 if

```
if (izraz)
    iskaz1
else
    iskaz2
```

Primer 1.13.1 *Program ilustruje if i ispisuje ukoliko je uneti ceo broj negativan*

```
#include <stdio.h>

main()
{
    int b;
    printf("Unesi ceo broj:");
    scanf("%d", &b);
```

```
    if (b < 0)
        printf("Broj je negativan\n");
}
```

Else se odnosi na prvi neuparen if, voditi o tome računa, ako želimo drugačije moramo da navedemo vitičaste zagrade.

```
if (izraz)
    if (izraz1) iskaz 1
else iskaz
```

ovo else se odnosi na drugo if a ne na prvo if!

```
if (izraz)
    {
    if (izraz1) iskaz 1
    }
else iskaz
```

tek sada se else odnosi na prvo if!!!

1.13.1 Else-if

```
if (izraz1)
    iskaz1
else if (izraz2)
    iskaz2
else if (izraz3)
    iskaz3
else if (izraz4)
    iskaz4
else iskaz

npr
if (a<5)
    printf("A je manje od 5\n");
else if (a==5)
    printf("A je jednako 5\n");
else if (a>10)
    printf("A je vece od 10\n");
else if (a==10)
    printf("A je jednako 10\n");
else printf("A je vece od pet i manje od 10\n");
```

Primer 1.13.2 *Program ilustruje if-else konstrukciju i ispituje znak broja.*

```
#include <stdio.h>

main()
{
    int b;
    printf("Unesi ceo broj : ");
    scanf("%d", &b);
    if (b < 0)
        printf("Broj je negativan\n");
    else if (b == 0)
        printf("Broj je nula\n");
    else
        printf("Broj je pozitivan\n");
}

Ulaz:
Unesi ceo broj:-5
Izlaz:
Broj je negativan

Ulaz:
Unesi ceo broj:5
Izlaz:
Broj je pozitivan
```

Primer 1.13.3 *Pogresan program sa dodelom = umesto poredjenja ==.*

```
#include <stdio.h>

main()
{
    int b;
    printf("Unesi ceo broj : ");
    scanf("%d", &b);

    /* Obratiti paznju na = umesto == Analizirati rad programa*/
    if (b = 0)
        printf("Broj je nula\n");
    else if (b < 0)
        printf("Broj je negativan\n");
    else
        printf("Broj je pozitivan\n");
```

```
}

Ulaz:
Unesi ceo broj:-5
Izlaz:
Broj je pozitivan
```

1.14 Uslovni izraz

Slično kao `if`.

```
izraz1 ? izraz2 : izraz3

z = (a<b)? a : b; /*z=min(a,b)*/
max = (a>b)? a : b;
```

1.15 Switch

```
switch (iskaz) {
    case konstantan_izraz1: iskazi1
    case konstantan_izraz2: iskazi2
    ...
    default: iskazi
}
```

Primer 1.15.1 *Voditi računa o upotrebi break-a.*

```
#include <stdio.h> /*
 Upotreba switch-a
*/

main() {
    char x;
    scanf("%c",&x);

    switch (x)
    {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u': printf(" x je samoglasnik");
```

```
                break;
        case 'r': printf(" x je r");
                break;
        default:  printf(" x je suglasnik");
    }
}
```

Primer 1.15.2 *Ilustracija switch konstrukcije.*

```
#include<stdio.h>
main()
{
int n;
printf("Unesi paran broj manji od 10\n");
scanf("%d",&n);
switch(n) {
case 0:
      printf("Uneli ste nulu\n");
      break;
case 2:
      printf("Uneli ste dvojku\n");
      break;
case 4:
      printf("Uneli ste cetvorku\n");
      break;
case 6:
      printf("Uneli ste sesticu\n");
      break;
case 8:
      printf("Uneli ste osmicu\n");
      break;
defalut:
      printf("Uneli ste nesto sto nije paran broj\n");
}
}
Ulaz:
Unesi paran broj manji od 10
2
Izlaz:
Uneli ste dvojku
```

1.16 Kontrola toka — `while`, `do-while` i `for`

1.16.1 while

`while(uslov) { ... }`
Uslov u zagradi se testira i ako je ispunjen telo petlje se izvrsava. Zatim se uslov ponovo testira i ako je ispunjen ponovo se izvrsava telo petlje. I tako sve dok uslov ne bude ispunjen. Tada se izlazi iz petlje i nastavlja sa prvom sledecom naredbom u programu.

Ukoliko iza while sledi samo jedna naredba nema potrebe za zagradama.

```
while (i<j)
    i=2*i;
```

1.16.2 do-while

Ovo je slično paskalskom repeat-until izrazu.

```
do iskaz while (izraz)
```

Primer 1.16.1 *Program ilustruje petlju do-while.*

```
#include <stdio.h>

main()
{
    int x;

    x = 1;
    do
    {
        printf("x = %d\n",x);
        x++; /* x++ je isto kao i x=x+1 */
    } while (x<=10);
}
```

1.16.3 for

Primer 1.16.2 *Program ilustruje petlju - for.*

```
#include <stdio.h>

main()
{
    int x;
```

```
    /* Inicijalizacija; uslov; inkrementacija*/
    for (x = 1; x < 5; x++)
        printf("x = %d\n",x);

}
Izlaz:
1
2
3
4
```

1.16.4 Dvostruka `for` petlja

Primer 1.16.3 *Dvostruka for petlja*

```
#include<stdio.h>
int main()
{
int i,j;

for(i=1; i<=10; i++)
    {
    for(j=1; j<=10; j++)
        printf("%d * %d = %d\t", i, j, i*j);
    printf("\n");
    }
}
```

Primer 1.16.4 *Trostruka for petlja*

```
#include<stdio.h>
int main()
{
int i,j;

for(i=1; i<=10; i++)
    for(j=1; j<=10; j++)
        {
        for(k=1; k<=10; k++)
            printf("%d * %d * %d = %d\t", i, j, k, i*j*k);
        printf("\n");
        }
}
```

Koliko puta se izvrsi naredba štampanja
`printf("%d * %d * %d = %d\t", i, j, k, i*j*k);`