Osnovi programiranja
Programski jezik C
— Zadaci sa vežbi —

Milena Vujošević - Janičić 2005/2006[1]

[1]Neki primeri su preuzeti sa sajta www.matf.bg.ac.yu/~filip i www.matf.bg.ac.yu/~jelenagr. Tekstovi su uglavnom zasnovani na knjizi *Programski jezik C*, autora Kerninghan & Ritchie

Sadržaj

Predgovor

Ovo je prateći materijal za vežbe koje držim iz predmenta Osnovi programiranja. 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.yu/~filip/pp/0405/index.pl). Takođe korišćen je i materijal sa sajta koleginice Jelene Grmuše www.matf.bg.ac.yu/~jelenagr. 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.yu/~milena

1

Programski jezik C

1.1 Zdravo svete!

Primer 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 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

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

Postoje kvalifikatori signed i unsigned koji se odnose na označene i neoznačene cele brojeve. Npr.
signed char: -128 do 127
dok je
unsigned char: od 0 do 255.
Float, double i long double.

Primer 3 *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 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 4

```
#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 5 *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 6 *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
```

1.6 Aritmetički operatori

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

Asocijativnost sleva na desno, prioritet kao u matematici.

Primer 7 *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 8 *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);
}
```

1.7 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.

1.8 Inkrementacija i dekrementacija

Operatori ++ i --

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

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

Primer 9 *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
```

1.9 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 10 *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 11 *Ilustracija lenjog izračunavanja logičkih operatora.*

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>

/* Globalna promenljiva, vidljiva i iz funkcije main() i
   iz funkcije izracunaj*/
int b = 0;

/* Funkcija ispisuje da je pozvana i uvecava promenjivu b.
   Funkcija uvek vraca vrednost 1 (tacno)
*/
int izracunaj()
{
    printf("Pozvano izracunaj()\n");
    b++;
    return 1;
}
```

```
main()
{
/* Funkcija izracunaj() ce biti pozvana
   samo za parne vrednosti a */
int a;
for (a = 0; a < 10; a++)
    if (a%2 == 0 && izracunaj())
        printf("Uslov ispunjen : a = %d, b = %d\n", a, b);
    else
        printf("Uslov nije ispunjen : a = %d, b = %d\n", a, b);

printf("----------------------------\n");

/* Funkcija izracunaj() ce se pozivati samo
   za neparne vrednosti a */
b = 0;
for (a = 0; a < 10; a++)
    if (a%2 == 0 || izracunaj())
        printf("Uslov ispunjen : a = %d, b = %d\n", a, b);
    else
        printf("Uslov nije ispunjen : a = %d, b = %d\n", a, b);
}
```

1.10 Kontrola toka — if, while, do - while, for

1.10.1 if

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

Primer 12 *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!!!

1.10.2 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 13 *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 14 *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
```

1.10.3 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;
```

1.10.4 do-while

Ovo je slično paskalskom repeat-until izrazu.

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

Primer 15 *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);
}
```

1.10.5 for

Primer 16 *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
```

1.11 Switch

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

Primer 17 *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 18 *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
```

1.12 Uslovni izraz

Slično kao `if`.

```
izraz1 ? izraz2 : izraz3
```

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

1.13 Simboličke konstante

Primer 19 *Konverzija centimetara u inče - while petlja.*

```
#include <stdio.h>

/* Definicija simbolickih konstanti preko #define direktiva */
/* U fazi pretprocesiranja se vrsi doslovna zamena konstanti
   njihovim vrednostima */

#define POCETAK 0
#define KRAJ 20
#define KORAK 10

main()
{
    int a;
    a = POCETAK;
    while (a <= KRAJ)
    {
        printf("%d cm = %f in\n", a, a/2.54);
        a += KORAK; /* isto sto i a = a + KORAK; */
    }
}

Izlaz:
0 cm = 0.000000 in
10 cm = 3.937008 in
20 cm = 7.874016 in
```

Primer 20 *Konverzija centimetara u inče - for petlja.*

```
#include <stdio.h>
#define POCETAK 0
#define KRAJ 20
#define KORAK 10

main()
{
```

```
    int a;
    for  (a = POCETAK; a <= KRAJ; a += KORAK)
        printf("%d cm = %f in\n", a, a/2.54);
}

Izlaz:
0 cm = 0.000000 in
10 cm = 3.937008 in
20 cm = 7.874016 in
```

Zadatak 2 *Šta će biti ispisano nakon izvršavanja sledećeg programa?*

```
#include <stdio.h>
#define EURO 85.90
main()
{
   printf("4 eura ima vrednost %f dinara\n", 4*EURO);
   printf("1 euro ima vrednost %.0f dinara\n",EURO);
}
```

1.14 Enumeracija

Izvesna alternativa za define

```
enum boolean {NO, YES};
enum meseci {JAN = 1, FEB, MAR, APR, MAJ, JUN,
                JUL, AVG, SEP, OKT, NOV, DEC}
enum boje {CRVENA, ZELENA=5, PLAVA,
            LJUBICASTA=10, ZUTA, CRNA}

koriscenje:

int x=0;
boje b;

x=CRVENA+3; /*x ce biti jednako tri*/

b=ZELENA;
x=b+CRNA; /* 5 + 12=17*/

b=0; /*Greska, ovako ne moze!!!*/
```

1.15 Funkcije

Primer 21 *sum - najjednostavnija funkcija koja sabira dva broja*

```
/* Definicija funkcije */
int sum(int a, int b)
{
    int c;
    c = a + b;
    return c;
    /* Ovo je krace moglo da bude napisano
       kao return a+b; */
}

main()
{

    int c;
    /* Poziv funkcije */
    c = sum(3,5);
    printf("%d\n", c);

    /* Ovo smo krace mogli da napisemo kao
    printf("%d\n", sum(3,5)); */
}
```

Primer 22 *Deklaracija funkcije moze da stoji nezavisno od definicije funkcije. Deklaracija je neophodna u situacijama kada se definicija funkcije navodi nakon upotrebe date funkcije u kodu.*

```
/* Deklaracija funkcije*/
int zbir(int, int);

main()
{
    /* Poziv funkcije */
    printf("%d\n", zbir(3,5));
}

/* Definicija funkcije */
int zbir(int a, int b)
{
    return a+b;
}
```

Primer 23 *power - funkcija koja stepenuje realan broj na celobrojni izlozilac*

```
 #include <stdio.h>

/* stepenuje x^k tako sto k puta pomnozi x */
float power(float x, int k)
{
    int i;
    float rezultat = 1;
    for (i = 0; i<k; i++)
        rezultat*=x;

    return rezultat;
}
```

Primer 24 *Verzija koja radi i za negativne izlozioce*

```
float power(float x, int k)
{
    int i;
    float s = 1;
    int negativan = (k<0);
    float rezultat;

    if (negativan)
        k = -k;

    for (i = 0; i<k; i++)
        s*=x;

    rezultat = negativan ? 1.0/s : s;
    return rezultat;
}

main()
{
    /* Poziv funkcije */
    float s = power(2.0,8);
    printf("%f\n", s);
}
```

1.16 Nizovi

Deklaracija niza:

```
int niz[5]; /* niz od 5 elemenata tipa int*/
```

Pristupanje elementima niza:

```
niz[0] = 4;
niz[1] = 2 * niz[0];        /*niz[1] = 8*/
niz[2] = niz[0] * niz[1];   /*niz[2] = 32*/
niz[3] = 5;
niz[4] = 7;
```

Unos vrednosti elemenata niza sa tastature:

```
for(i=0; i<5; i++)
    scanf("%d", &a[i]);
```

Stampanje elemenata niza

```
for(i=0; i<5; i++)
    printf("%d ", a[i]);
```

Brojanje elemenata niza je od nule!
Pristupanje elementu niza, indeks može da bude proizvoljan izraz celobrojne vrednosti: `niz[i*2]=5`.

Primer 25 *Program ilustruje korišćenje nizova. Ispisuje 10 unetih brojeva unazad.*

```
#include <stdio.h>

main()
{
    int a[10];
    int i;
    for (i = 0; i<10; i++)
    {
        printf("a[%d]=",i);
        scanf("%d",&a[i]);
    }

    printf("Unazad : \n");

    for (i = 9; i>=0; i--)
        printf("a[%d]=%d\n",i,a[i]);
}
```

Primer 26 *Program pronalazi maksimum brojeva sa ulaza - verzija sa nizom.*

```
#include <stdio.h>
#define BR_ELEM 5
main()
{
    int a[BR_ELEM];
    int i;
    int max;

    /* Ucitavamo niz brojeva */
    for (i = 0; i < BR_ELEM; i++)
        scanf("%d",&a[i]);

    /* Pronalazimo maksimum */
    max = a[0];
    for (i = 1; i < BR_ELEM; i++)
        if (a[i]>max)
            max = a[i];

    /* Ispisujemo maksimum */
    printf("Max = %d\n",max);
}
```

Primer 27 *Program pronalazi maksimum brojeva sa ulaza - verzija bez niza.*

```
#include <stdio.h>
#define BR_ELEM 5

main()
{
    int a, max, i;
    scanf("%d",&a);
    max = a;
    for (i = 1; i < BR_ELEM; i++)
    {
        scanf("%d",&a);
        if (a>max)
            max = a;
    }

    printf("Max : %d\n", max);
```

```
}
```

1.17 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, tab i slično.

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

Konstantna niska: "Ja sam niska"
ili
"" /*prazna niska*/
Navodnici nisu deo niske već se koriste da bi je ograničili. Ako ih želimo unutar niske, oni se navode sa `\"`.

Konstantna niska je polje znakova. Da bi se znalo gde je kraj niske, fizičko memorisanje liste zahteva da postoji jedan znak više koji označava kraj, to je `'\0'`. Da bi se odredila dužina niske mora se proći kroz celu nisku.

!!!Važno:

Koja je razlika između "x" i 'x'?

Primer 28 *Primer funkcije koja izračunava dužinu niske znakova.*

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

```
int strlen(char s[])
{
    int i=0;

    while (s[i] != '\0')
        ++i;
    return i;
}

int main()
{
printf("Duzina ove niske
        je: %d \n",strlen("Duzina ove niske je:"));
return 0;
}
```

1.18 Konverzija

1.18.1 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*/
```

1.18.2 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 29 *Kako izbeći celobrojno deljenje*

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

1.18.3 Funkcije koje vrše konverziju

Primer 30

```
#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 31 *Funkcija koja konvertuje velika slova u mala slova.*

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

```
/* Konvertuje karakter iz velikog u malo slovo */
char lower(char c)
{
if (c >= 'A' && c <= 'Z')
    return c - 'A' + 'a' ;
else
    return c;
}

main()
{
char c;
printf("Unesi neko veliko slovo:\n");
scanf("%c", &c);
printf("Odgovarajuce malo slovo je %c\n", lower(c));
}
```

Primer 32 *Konvertovanje niske cifara u ceo broj.*

```
#include<stdio.h>

/* atoi:  konvertuje s u ceo broj */
int atoi(char s[])
{
int i, n;
n = 0;
for (i = 0; (s[i] >= '0') && (s[i] <= '9'); ++i)
    n = 10 * n + (s[i] - '0');
return n;
}

main()
{
int n;
n = atoi("234");
printf("\nN je : %d\n",n);
}
```

1.19 Operator `sizeof()`

Primer 33 *Demonstracija sizeof operatora.* ***`Sizeof`*** *operator izračunava veličinu tipa odnosno promenjive.*

```
#include<stdio.h>
main()
{
int i;
float f;
int n[10];

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));
printf("sizeof(n)=%d\n", sizeof(n));
printf("Broj elemenata niza n : %d\n", sizeof(n)/sizeof(int));

}

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
sizeof(n)=40
Broj elemenata niza n : 10
```

1.20 Znakovni ulaz i izlaz

Funkcija za čitanje jednog znaka sa ulaza
`c = getchar()`
promenljiva c sadrži jedan znak sa ulaza.

Funkcija za štampanje jednog znaka na izlaz
`putchar(c)`
štampa sadržaj promenljive c obično na ekranu.

Konstanta `EOF` je celobrojna vrednost definisana u biblioteci `<stdio.h>`. Ovu vrednost vrati funkcija `getchar()` kada nema više ulaza. Nazvana je `EOF` kao `End Of File`, kraj datoteke. Ova vrednost mora da se razlikuje od svake vrednosti koja može da bude karakter. Zato za `c` za koje je `c=getchar()` treba da koristimo tip dovoljno veliki da moze da prihvati sve sto moze da vrati `getchar()`, dakle i `EOF`. Zbog toga se za `c` koristi tip `int`.

Primer 34 *Program cita jedan karakter i ispisuje ga - demonstracija putchar i getchar.*

```
#include <stdio.h>

main()
{
    int c;          /* Karakter - obratiti paznju na int */
    c = getchar(); /* cita karakter sa standardnog ulaza */
    putchar(c);    /* pise karakter c na standardni izlaz */

    putchar('\n'); /* prelazak u novi red */
    putchar('a');  /* ispisuje malo a */
    putchar(97);   /* ekvivalentno prethodnom */
}
Ulaz:
s
Izlaz iz programa:
s
s
aa
```

Primer 35 *Program prepisuje standardni ulaz na standardni izlaz. Ilustracija redirekcije standardnog ulaza i izlaza, pokrenuti program sa :*

```
./a.out <primer.c
./a.out >tekst.txt
./a.out <primer.c >kopija.c
```

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

```
main()
{
/*Koristi se int a ne char zato sto zelimo
  da razlikujemo kraj ulaza od vazecih znakova*/
int c;

/*Ucitava se prvi znak sa ulaza*/
c = getchar();

/*EOF predstavlja celobrojnu vrednost kraja datoteke.
  To je konstanta definisana u <stdio.h>*/
while (c != EOF) {
    putchar(c);
    c = getchar();
    }
}
```

Bilo koje dodeljivanje vrednosti je izraz koji ima vrednost a to je vrednost leve strane posle dodeljivanja.

Primer 36 *Program koji kopira ulaz na izlaz, skraćeni kod.*

```
#include <stdio.h>

main()
{
    int c;

    /* Obratiti paznju na raspored zagrada */
    while ((c = getchar()) != EOF)
        putchar(c);
}
```

Primer 37 *Brojanje znakova na ulazu.*

```
#include <stdio.h>

main()
{
    long nc;

    nc = 0;
    while (getchar() != EOF)
        ++nc;
    /* %ld odnosi se na tip long. */
```

```
    printf("%ld\n", nc);
}
```

Primer 38 *Brojanje znakova na ulazu korišćenjem for petlje.*

```
#include <stdio.h>

  main()
   {
       double nc;

        /*For petlja mora da ima telo pa makar
        ono bilo przno*/
       for (nc = 0; getchar() != EOF; ++nc)
           ;
       printf("%.0f\n", nc);
   }
```

Primer 39 *Program broji linije i znakove na ulazu.*

```
#include <stdio.h>
main()
{
int znak; /*prihvata znak sa ulaza */
long linije=0 ; /*brojac linija */
long br_znak=0; /*brojac znakova na ulazu */

while ( (znak=getchar() ) != EOF)
    {
    br_znak++;
    if (znak=='\n') linije ++;
    }

printf("Prelazaka u novi red: %ld, karaktera: %ld \n",linije,br_znak);
}
```

Primer 40 *Program broji blankove, horizontalne tabulatore i linije na ulazu.*

```
#include <stdio.h>
main()
{
int znak;       /*prihvata znak sa ulaza */
int Blanks=0;   /*brojac blankova */
int Tabs=0;     /*brojac horizontalnih tabulatora */
```

```
int NewLines=0; /*brojac linija */

/*UOCITI: blok naredbi while ciklusa NIJE OGRADJEN
  viticastim zagradama jer postoji samo jedna if naredba! */
while( (znak=getchar())!=EOF )
    if( znak==' ' ) ++Blanks; /* brojimo blanko simbole */
    else if( znak=='\t' ) ++Tabs; /* brojimo tab-ove */
        else if( znak=='\n' ) ++NewLines; /* brojimo redove */

/*izdavanje rezultata na standardni izlaz*/
printf("Blankova: %d. Tabulatora: %d. Prelazaka u novi red: %d\n",
                                                Blanks, Tabs, NewLines);

}
```

Primer 41 *Program prepisuje ulaz na izlaz pri čemu više blanko znakova zamenjuje jednim.*

```
#include <stdio.h>

main()
{
   int znak; /*tekuci znak sa ulaza*/
   int preth; /*znak koji prethodi tekucem */
   preth='a'; /* inicijalizujemo vrednost prethodnog
                 da bi prvi prolazak kroz petlju bio ispravan*/
     while ( (znak=getchar() ) !=EOF)
       {
         if (znak !=' ' || preth != ' ') putchar(znak);
            preth=znak;
        }
}
```

Primer 42 *Program vrši prebrojavanje cifara unetih na ulazu.*

```
#include <stdio.h>

main()
{
    int c;
    int br_cifara = 0;
    while ((c = getchar()) != EOF)
        if ('0'<=c && c<='9')
            br_cifara++;
```

```
    printf("Broj cifara je : %d\n", br_cifara);
}
```

Primer 43 *Program vrši brojanje pojavljivanja karaktera 0, 1 i 2 (ilustruje switch).*

```
#include <stdio.h>

main()
{
    int c;
    int br_0=0, br_1=0, br_2=0;

    while ((c = getchar()) != EOF)
    {
        switch(c)
        {
            /* Obratiti paznju da nije
                case 0: */
            case '0':
                br_0++;
                break;  /* Isprobati veziju bez break */
            case '1':
                br_1++;
                break;
            case '2':
                br_2++;
                break;
            default:
        }
    }
    printf("Br 0 : %d\nBr 1 : %d\nBr 2 : %d\n",br_0, br_1, br_2);
}
```

1.21 Nizovi

Primer 44 *Program ilustruje inicijalizaciju nizova.*

```
#include <stdio.h>

main()
{
    /* Niz inicijalizujemo tako sto mu navodimo vrednosti
```

```
       u viticasnim zagradama. Dimenzija niza se odredjuje
       na osnovu broja inicijalizatora */
    int a[] = {1, 2, 3, 4, 5, 6};

    /* Isto vazi i za niske karaktera */
    char s[] = {'a', 'b', 'c'};

    /* Ekvivalentno prethodnom bi bilo
    char s[] = {97, 98, 99};
    */

    /* Broj elemenata niza */
    int a_br_elem = sizeof(a)/sizeof(int);
    int s_br_elem = sizeof(s)/sizeof(char);

    /* Ispisujemo nizove */

    int i;
    for (i = 0; i < a_br_elem; i++)
        printf("a[%d]=%d\n",i, a[i]);


    for (i = 0; i < s_br_elem; i++)
        printf("s[%d]=%c\n",i, s[i]);

}
```

Primer 45 *Program uvodi niske karaktera terminisane nulom.*

```
#include <stdio.h>

main()
{
    /* Poslednji bajt niske karaktera se postavlja na '\0' tj. 0 */
    char s[] = {'a', 'b', 'c', '\0' };

    /* Kraci nacin da se postigne prethodno */
    char t[] = "abc";

    /* Ispis niske s karakter po karakter*/
    int i;
    for (i = 0; s[i] != '\0'; i++)
        putchar(s[i]);
    putchar('\n');
```

```
    /* Ispis niske s koristeci funkciju printf */
    printf("%s\n", s);

    /* Ispis niske t karakter po karakter*/
    for (i = 0; t[i] != '\0'; i++)
        putchar(t[i]);
    putchar('\n');

    /* Ispis niske t koristeci funkciju printf */
    printf("%s\n", t);
}
```

Primer 46 *Brojanje pojavljivanja svake od cifara. Koriscenje niza brojača.*

```
#include <stdio.h>

/* zbog funkcije isdigit */
#include <ctype.h>

main()
{
    /* Niz brojaca za svaku od cifara */
    int br_cifara[10];
    int i, c;

    /* Resetovanje brojaca */
    for (i = 0; i < 10; i++)
        br_cifara[i] = 0;

    /* Citamo sa ulaza i povecavamo odgovarajuce brojace */
    while ((c = getchar()) != EOF)
        if (isdigit(c))
            br_cifara[c-'0']++;

    /* Ispis rezultata */
    for (i = 0; i < 10; i++)
        printf("Cifra %d se pojavila %d put%s\n",
            i, br_cifara[i], br_cifara[i] == 1 ? "" : "a");
}
```

Primer 47 *Brojanje reči*

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

```
#define IN   1  /* inside a word */
#define OUT  0  /* outside a
word */

/* count lines, words, and characters in input */

main()
{
int c, nl, nw, nc, state;

state = OUT;

/*Postavljaju se sve tri promenljive na nulu*/
/*Isto kao da smo napisali
nl = (nw = (nc = 0));*/

nl = nw = nc = 0;
while ((c = getchar()) != EOF) {
    ++nc;
    if (c == '\n')
        ++nl;
    /*Operator || znaci OR*/
    if (c == ' ' || c == '\n' || c == '\t')
        state = OUT;
    else if (state == OUT) {
        state = IN;
        ++nw;
                }
}
printf("%d %d %d\n", nl, nw, nc);
}
```

1.22 Dvostruka `for` petlja

Primer 48 *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 49 *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);`

1.23 Formiranje HTML dokumenta

Primer 50 *Prilikm pokretanja programa koristiti redirekciju:*
`a.out >primer.html`
kako bi se rezultat rada programa upisao u datoteku `primer.html`.

```
/*Ovaj program formira html dokument*/
#include <stdio.h>
main()
{
printf("<html><head><title>Ova stranica
    je napravljena u c-u</title></head>");
printf("<body><h3 align=center>
            Rezultat </h3></body></html>");
}
```

Primer 51 *Napisati program koji generiše html dokument sa engleskim alfabetom.*

```
#include <stdio.h>
main()
{
int i;
printf("<HTML><head><title>Engleski alfabet</title><head>\n");
printf("<body><ul>");
for(i=0;i<=25;i++)
    printf("<li> %c %c \n",'A'+i,'a'+i);
printf("</ul></body></HTML>\n"); }
```

Primer 52 *Napisati program koji generise html dokument koji prikazuje tablicu mnozenja za brojeve od 1 do 10.*

```
#include<stdio.h>
main()
{
int i,j;
printf("<html><head><title>Mnozenje</title></head>");
printf("<body><h3 align=center> Rezultat </h3>");
printf("<table border=1>\n");

/* Prva vrsta sadrzi brojeve od 1 do 10*/
printf("<tr>");
printf("<th></th>");
for(i=1; i<=10; i++)
    printf("<th> %d </th>\n", i);
printf("</tr>");

for(i=1; i<=10; i++)
    {
    printf("<tr>");

    /* Na pocetku svake vrste stampamo broj
    odgovarajuce vrste*/
    printf("<th>%d</th>", i);

    for(j=1; j<=10; j++)
        printf("<td>%d\t</td>\n", i*j);

    printf("</tr>");
}
printf("</table>");
printf("</body></html>");
}
```

1.24 Funkcije — prenos parametara po vrednosti

Primer 53 *Funkcija koja proverava da li je broj prost i program koji ispisuje sve proste brojeve manje od 100.*

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

int prost(int p)
{
    int i, koren, ind;
    koren=sqrt(p);
    ind=(p%2) || (p==2);
    i=3;
    while (ind && i<=koren)
        {
        ind=p%i;
        i+=2;
        }
    return ind;
}

main()
{
int k;
for(k=2;k<=100;k++)
    if (prost(k)) printf("%d ",k);
}
```

Primer 54

```
#include <stdio.h> /* Ilustruje vidljivost imena*/

int i=10;

void main() {
    {
        int i=3;
        {
            int i=1;
            printf("%d\n", i);
        }
        printf("%d\n",i);
    }
```

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

Primer 55 *Demonstracija prenosa parametara po vrednosti - preneti parametri se ne mogu menjati*

```
#include <stdio.h>
void f(int x)
{
x++;
}

main()
{
int x=3;
f(x);
printf("%d\n", x);
}
```

Primer 56 *Demonstrira prenos nizova u funkciju - preneti niz se moze menjati.*

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

/* Funkcija ucitava rec sa standardnog ulaza i smesta je u niz karaktera s.
   Ovo uspeva zbog toga sto se po vrednosti prenosi adresa pocetka niza,
   a ne ceo niz */
void get_word(char s[])
{
    int c, i = 0;
    while (!isspace(c=getchar()))
        s[i++] = c;
    s[i] = '\0';
}

main()
{
    /* Obavezno je alocirati memoriju za niz karaktera */
    char s[100];

    get_word(s);
    printf("%s\n", s);
}
```

Primer 57 *Funkcija za ispis niza brojeva - demonstrira prenos nizova brojeva u funkciju.*

```
#include <stdio.h>

/* Nizovi se prenose tako sto se prenese adresa njihovog pocetka.
   Uglaste zagrade ostaju prazne!

   Nizove je neophodno prenositi zajedno sa dimenzijom niza
   (osim za niske karaktera, jer tamo vazi konvencija da
   se kraj niza obelezava znakom '\0')
*/
void print_array(int a[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ",a[i]);
    putchar('\n');

    /* Obratite paznju na ovo : */
    printf("sizeof(a) - u okviru fje : %d\n", sizeof(a));
}

main()
{
    int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};

    printf("sizeof(a) - u okviru main : %d\n", sizeof(a));
    print_array(a, sizeof(a)/sizeof(int));
}
```

Primer 58 *Funkcija za ispis niske karaktera - demonstrira prenos niske karaktera u funkciju.*

```
#include <stdio.h>

/* Uz nisku karaktera nije potrebno prenositi dimenziju
   ukoliko se postuje dogovor
   da se svaka niska zavrsava karakterom '\0'.*/
void print_string(char s[])
{
    int i;
    for (i = 0; s[i]!='\0'; i++)
        putchar(s[i]);
```

```
}


main()
{
    print_string("Zdravo\n");
}
```

1.25 Break i continue

Primer 59 *Funkcija uklanja beline, tabulatore ili znak za kraj reda sa kraja stringa.*

```
int trim(char s[])
{
int n;
for (n = strlen(s)-1; n >= 0; n--)
    if (s[n] != ' ' && s[n] != '\t' && s[n] != '\n')
                break;
s[n+1] = '\0';
return n;
}
```

`Continue` se ređe koristi, on prouzrokuje da se pređe na sledeću iteraciju u petlji.

Primer 60

```
for(i=0; i<n; i++)
{
if (a[i]==0) continue;
... /* obradi pozitivne elemente nekako*/
}
```

1.26 Rad sa niskama karaktera

Primer 61 *string_reverse - obrće nisku karaktera.*

```
#include <stdio.h>

/* Ova funkcija racuna duzinu date niske karaktera.
   Umesto nje, moguce je koristiti standardnu funkciju strlen
   za cije je koriscenje potrebno ukljuciti zaglavlje
   <string.h>
 */
```

```
int string_length(char s[])
{
    int i;
    for (i = 0; s[i]!='\0'; i++)
        ;

    return i;
}

/* Funkcija obrce nisku karaktera */
void string_reverse(char s[])
{
    int i, j;
    for (i = 0, j = string_length(s)-1; i<j; i++, j--)
    {
        int tmp = s[i];
        s[i] = s[j];
        s[j] = tmp;
    }

    /* Napomena : razlikovati prethodnu petlju od dve ugnjezdjene petlje
    for ( i = 0; ....)
        for ( j = duzina(s)-1; ...
     */

}

main()
{
    char s[] = "Zdravo svima";
    string_reverse(s);
    printf("%s\n", s);
}
/*
Izlaz:
amivs ovardZ
*/
```

Primer 62 `strlen`, `strcpy`, `strcat`, `strcmp`, `strchr`, `strstr` - *manipulacija niskama karaktera. Vezbe radi, implementirane su funkcije biblioteke* `string.h`

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

```
/* Izracunava duzinu stringa */
int string_length(char s[])
{
    int i;
    /* Uslov s[i] je ekvivalentan uslovu
       s[i]!='\0' ili uslovu s[i]! = 0*/
    for (i = 0; s[i]; i++)
        ;
    return i;
}


/* Kopira string src u string dest.
  Pretpostavlja da u dest ima dovoljno prostora. */
void string_copy(char dest[], char src[])
{
    /* Kopira karakter po karakter, sve dok nije iskopiran karakter '\0' */
    int i;
    for (i = 0; (dest[i]=src[i]) != '\0'; i++)
        ;

    /* Uslov != '\0' se, naravno, moze izostaviti :

    for (i = 0; dest[i]=src[i]; i++)
        ;
    */

}

/* Nadovezuje string t na kraj stringa s.
   Pretpostavlja da u s ima dovoljno prostora. */
void string_concatenate(char s[], char t[])
{
    int i, j;
    /* Pronalazimo kraj stringa s */
    for (i = 0; s[i]; i++)
        ;

    /* Vrsi se kopiranje, slicno funkciji string_copy */
    for (j = 0; s[i] = t[j]; j++, i++)
        ;
}
```

```
/* Vrsi leksikografsko poredjenje dva stringa.
    Vraca :
        0 - ukoliko su stringovi jednaki
        <0 - ukoliko je s leksikografski ispred t
        >0 - ukoliko je s leksikografski iza t
*/
int string_compare(char s[], char t[])
{
    /* Petlja tece sve dok ne naidjemo na prvi razliciti karakter */
    int i;
    for (i = 0; s[i]==t[i]; i++)
        if (s[i] == '\0') /* Naisli smo na kraj oba stringa,
                             a nismo nasli razliku */
            return 0;

    /* s[i] i t[i] su prvi karakteri u kojima se niske razlikuju.
       Na osnovu njihovog odnosa, odredjuje se odnos stringova */
    return s[i] - t[i];
}

/* Pronalazi prvu poziciju karaktera c u stringu s, odnosno -1
   ukoliko s ne sadrzi c */
int string_char(char s[], char c)
{
    int i;
    for (i = 0; s[i]; i++)
        if (s[i] == c)
            return i;
        /* nikako
        else
            return -1;
        */
    /* Nije nadjeno */
    return -1;
}

/* Pronalazi poslednju poziciju karaktera c u stringu s, odnosno -1
   ukoliko s ne sadrzi c */
int string_last_char(char s[], char c)
{
    /* Pronalazimo kraj stringa s */
    int i;
```

```
    for (i = 0; s[i]; i++)
        ;

    /* Krecemo od kraja i trazimo c unazad */
    for (i--; i>=0; i--)
        if (s[i] == c)
            return i;

    /* Nije nadjeno */
    return -1;

    /*
    Koristeci string_length :

    for (i = string_length(s) - 1; i>0; i--)
        if (s[i] == c)
            return i;

    return -1;
    */
}

/* Proverava da li string str sadrzi string sub.
   Vraca poziciju na kojoj sub pocinje, odnosno -1 ukoliko ga nema
*/
int string_string(char str[], char sub[])
{
    int i, j;
    /* Proveravamo da li sub pocinje na svakoj poziciji i */
    for (i = 0; str[i]; i++)
        /* Poredimo sub sa str pocevsi od poziciji i
           sve dok ne naidjemo na razliku */
        for (j = 0; str[i+j] == sub[j]; j++)
            /* Nismo naisli na razliku a ispitali smo
               sve karaktere niske sub */
            if (sub[j+1]=='\0')
                return i;
    /* Nije nadjeno */
    return -1;
}

main()
{
```

```
    char s[100];
    char t[] = "Zdravo";
    char u[] = " svima";

    string_copy(s, t);
    printf("%s\n", s);

    string_concatenate(s, u);
    printf("%s\n", s);

    printf("%d\n",string_char("racunari", 'n'));
    printf("%d\n",string_last_char("racunari", 'a'));

    printf("%d\n",string_string("racunari", "rac"));
    printf("%d\n",string_string("racunari", "ari"));
    printf("%d\n",string_string("racunari", "cun"));
    printf("%d\n",string_string("racunari", "cna"));
}

/*
Izlaz:
Zdravo
Zdravo svima
4
5
0
5
2
-1*/
```

1.27 Makroi

```
#define ime tekst_zamene
```

Zamene se vrše samo na simbolima, a ne obavljaju se u niskama nutar navodnika.

Moguće je definisati makroe sa argumentima tako da tekst zamene bude različit za različita pojavljivanja makroa.

Primer 63 *Demonstracija pretprocesorske direktive* `#define`

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

```
/* Racuna sumu dva broja */
#define sum(a,b) ((a)+(b))

/* Racuna kvadrat broja - pogresna verzija */
#define square_w(a) a*a

/* Racuna kvadrat broja */
#define square(a) ((a)*(a))

/* Racuna minimum tri broja */
#define min(a, b, c) (a)<(b)? ((a)<(c)? (a) : (c)) : ((b)<(c)? (b) : (c))

main()
{
    printf("sum(3,5) = %d\n", sum(3,5));
    printf("square_w(5) = %d\n", square_w(5));
    printf("square_w(3+2) = %d\n", square_w(3+2));
    printf("square(3+2) = %d\n", square(3+2));
    printf("min(1,2,3) = %d\n", min(1,2,3));
    printf("min(1,3,2) = %d\n", min(1,3,2));
    printf("min(2,1,3) = %d\n", min(2,1,3));
    printf("min(2,3,1) = %d\n", min(2,3,1));
    printf("min(3,1,2) = %d\n", min(3,1,2));
    printf("min(3,2,1) = %d\n", min(3,2,1));
}

Izlaz iz programa:
sum(3,5) = 8
square_w(5) = 25
square_w(3+2) = 11
square(3+2) = 25
min(1,2,3) = 1
min(1,3,2) = 1
min(2,1,3) = 1
min(2,3,1) = 1
min(3,1,2) = 1
min(3,2,1) = 1
```

Primer 64

```
#define max(A, B) ((A)>(B) ? (A) : (B))
```

na osnovu ovoga će linija

```
x=max(p+q, r+s)
```

biti zamenjena linijom

```
x=((p+q) > (r+s) ? (p+q) : (r+s));
```

Treba voditi računa o sporednim efektima. Sledeća linija koda pruzrokovaće uvećanje vrednosti `i` ili `j` za dva.

```
max(i++, j++)
```

Takođe treba voditi računa o zagradama. Sledeći makro prouzrokovaće neočekivane rezultate za upotrebu `square(a+1)`

```
#define square(x) x*x
```

Primer 65 *Ilustacija beskonačne petlje:*

```
#define forever for(;;);
```

Primer 66

```
#include  <stdio.h>
#define max1(x,y) (x>y?x:y)
#define max2(x,y) ((x)>(y)?(x):(y))
#define swapint(x,y) { int z; z=x;  x=y; y=z; }
#define swap(t,x,y) { \
    t z; \
    z=x;  \
    x=y; \
    y=z; }

main()
{

int x=2,y=3;

printf( "max1(x,y) = %d\n", max1(x,y)  );
/* max1(x,y) = 3 */

/* Zamena makroom se ne vrsi
unutar niski pod navodnicima*/
printf( "max1(x=5,y) = %d\n", max1(x,y)  );
/* max1(x=5,y) = 3 */

printf( "max1(x++,y++) = %d\n", max1(x++,y++)  );
/* max1(x++,y++) = 4 */

printf( "x = %d, y = %d\n", x, y  );
```

```
/* x = 3, y = 5 */

swapint(x,y);

printf( "x = %d, y = %d\n", x, y  );
/* x = 5, y = 3 */

swap(int,x,y);
printf( "x = %d, y = %d\n", x, y  );
/* x = 3, y = 5 */
}

Izlaz:
max1(x,y) = 3
max1(x=5,y) = 3
max1(x++,y++) = 4
x = 3, y = 5
x = 5, y = 3
x = 3, y = 5
```

1.28 Bitski operatori

!!!Ne mešati sa logičkim operatorima!!!

```
& bitsko AND
| bitsko OR
^ bitsko ekskluzivno OR
<< levo pomeranje
>> desno pomeranje
~ jedinicni komplement
```

Primer 67 *Demonstracija bitskih operatora*

```
#include <stdio.h>

main()
{  printf("%o %o\n",255,15);
   printf( "255 & 15 = %d\n", 255 & 15 );
   printf( "255 | 15 = %d\n", 255 | 15 );
   printf( "255 ^ 15 = %d\n", 255 ^ 15 );
   printf( "2 << 2   = %d\n", 2 << 2 );
   printf( "16 >> 2  = %d\n", 16 >> 2 );
}
```

```
Izlaz iz programa je:
377 17
255 & 15 = 15
255 | 15 = 255
255 ^ 15 = 240
2 << 2   = 8
16 >> 2  = 4
```

Primer 68 *print_bits - stampa bitove u zapisu datog celog broja x.*

```
#include <stdio.h>

/* Funkcija stampa bitove datog celog broja x.
   Vrednost bita na poziciji i je 0 ako i
   samo ako se pri konjunkciji broja x sa maskom
   000..010....000 - sve 0 osim 1 na poziciji i,
   dobija 0.
   Funkcija krece od pozicije najvece tezine
   kreirajuci masku pomeranjem jedinice u levo
   za duzina(x) - 1 mesto, i zatim pomerajuci
   ovu masku za jedno mesto u levo u svakoj
   sledecoj iteraciji sve dok maska ne postane 0.
*/

void print_bits(int x)
{
    /* Broj bitova tipa unsigned */
    int wl = sizeof(int)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

main()
{
    print_bits(127);
    print_bits(128);
    print_bits(0x00FF00FF);
    print_bits(0xFFFFFFFF);
}
```

```
Izlaz iz programa:
00000000000000000000000001111111
00000000000000000000000010000000
00000000111111110000000011111111
11111111111111111111111111111111
```

Primer 69 *Program proverava da li se na k-tom mestu nalazi 1.*

```
#include <stdio.h>

/* Pozicije brojimo kao u sledecem primeru:
   poz: ... 10 9 8 7 6 5 4 3 2 1 0
   bit: ...  0 0 0 1 1 1 0 0 1 1 0  */

main()
{
   int n,k;
   printf("Unesite broj i poziciju tog broja koju zelite da proverite:\n");
   scanf("%d%d",&n,&k);
   if ((n & (1 << k))!=0)
     printf("Bit je 1\n");
   else
     printf("Bit je 0\n");
}
```

Primer 70 *Program postavlja na k-to mesto 1*

```
#include <stdio.h>

void print_bits(int x)
{
    /* Broj bitova tipa unsigned */
    int wl = sizeof(int)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

main(){
  int n,k;
```

```
  printf("Unesite broj i poziciju tog broja koju zelite da promenite:\n")
  scanf("%d%d",&n,&k);
  printf("Binarno, une\v seni broj je\n");
  print_bits(n);
  printf("Novi broj je %d\n",(n |(1<<k)));
  printf("Binarno, novi broj je\n");
  print_bits((n |(1<<k)));
}
```

Izrazom `a>>b` vrši se pomeranje sadržaja operanda `a` predstavljenog u binarnom obliku za `b` mesta u desno. Popunjavanje upražnjenih mesta na levoj strani zavisi od tipa podataka i vrste računara. Ako se pomeranje primenjuje nad operandom tipa `unsigned` popunjavanje je nulama. Ako se radi o označenom operandu popunjavanje je jedinicama kada je u krajnjem levom bitu jedinica, a nulama kada je u krajnjem levom bitu nula.

Primer 71 *sum_of_bits - izračunava sumu bitova datog neoznačenog broja.*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');

    putchar('\n');
}

int sum_of_bits(unsigned x)
{
    int br;
    for (br = 0; x; x>>=1)
        if (x&1)
            br++;

    return br;
}
```

```
main()
{
    printf("Binarni zapis broja 127 je\n");
    print_bits(127);
    printf("Suma bitova broja 127 je %d\n",sum_of_bits(127));
    printf("Binarni zapis broja 128 je\n");
    print_bits(128);
    printf("Suma bitova broja 128 je %d\n",sum_of_bits(128));
    printf("Binarni zapis broja 0x00FF00FF je\n");
    print_bits(0x00FF00FF);
    printf("Suma bitova broja 0x00FF00FF je %d\n",sum_of_bits(0x00FF00FF));
    printf("Binarni zapis broja 0xFFFFFFFF je\n");
    print_bits(0xFFFFFFFF);
    printf("Suma bitova broja 0xFFFFFFFF je %d\n",sum_of_bits(0xFFFFFFFF));
}

Izlaz iz programa:
Binarni zapis broja 127 je
00000000000000000000000001111111
Suma bitova broja 127 je 7
Binarni zapis broja 128 je
00000000000000000000000010000000
Suma bitova broja 128 je 1
Binarni zapis broja 0x00FF00FF je
00000000111111110000000011111111
Suma bitova broja 0x00FF00FF je 16
Binarni zapis broja 0xFFFFFFFF je
11111111111111111111111111111111
Suma bitova broja 0xFFFFFFFF je 32
```

Primer 72 *get_bits, set_bits, invert_bits - izdvajanje, postavljanje i invertovanje pojedinacnih bitova*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
    int wl = sizeof(unsigned)*8;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask >>= 1)
        putchar(x&mask ? '1' : '0');
```

```
    putchar('\n');
}

/* Funkcija vraca n bitova broja x koji pocinju na poziciji p */
unsigned get_bits(unsigned x, int p, int n)
{
    /* Gradimo masku koja ima poslednjih n jedinica
        0000000...00011111
       tako sto sve jedinice ~0 pomerimo u levo za n mesta
            1111111...1100000
       a zatim komplementiramo
    */
    unsigned last_n_1 = ~(~0 << n);

    /* x pomerimo u desno za odgovarajuci broj mesta, a zatim
       konjunkcijom sa konstruisanom maskom obrisemo pocetne cifre */

    return (x >> p+1-n) & last_n_1;
}

/* Funkcija vraca modifikovano x tako sto mu je izmenjeno n bitova
   pocevsi od pozicije p i na ta mesta je upisano poslednjih n bitova
   broja y */
unsigned set_bits(unsigned x, int p, int n, unsigned y)
{
    /* Maska 000000...000111111  - poslednjih n jedinica */
    unsigned last_n_1 = ~(~0 << n);

    /* Maska 1111100..000111111 - n nula pocevsi od pozicije p */
    unsigned middle_n_0 = ~(last_n_1 << p+1-n);

    /* Brisemo n bitova pocevsi od pozicije p */
    x = x & middle_n_0;

    /* Izdvajamo poslednjih n bitova broja y i pomeramo ih
       na poziciju p */
    y = (y & last_n_1) << p+1-n;

    /* Upisujemo bitove broja y u broj x i vracamo rezultat */
    return x | y;
}

/* Invertuje n bitova broja x pocevsi od pozicije p */
```

```
unsigned invert_bits(unsigned x, int p, int n)
{
    /* Maska 000000111...1100000 - n jedinica pocevsi od pozicije p */
    unsigned middle_n_1 = ~(~0 << n) << p+1-n;

    /* Invertujemo koristeci ekskluzivnu disjunkciju */
    return x ^ middle_n_1;
}

main()
{
    unsigned x = 0x0AA0AFA0;
    print_bits(x);

    print_bits(get_bits(x, 15, 8));
    print_bits(set_bits(x, 15, 8, 0xFF));
    print_bits(invert_bits(x, 15, 8));
}

Izlaz iz programa:
00001010101000001010111110100000
00000000000000000000000010101111
00001010101000001111111110100000
00001010101000000101000010100000
```

Primer 73 *right_rotate_bits, mirror_bits - rotiranje i simetrija bitova.*

```
#include <stdio.h>

/* Pomocna funkcija - stampa bitove neoznacenog broja */
void print_bits(unsigned x)
{
int wl = sizeof(unsigned)*8;

unsigned mask;
for (mask = 1<<wl-1; mask; mask >>= 1)
    putchar(x&mask ? '1' : '0');

putchar('\n');
}

/* Funkcija vrsi rotaciju neoznacenog broja x za n pozicija u desno */
unsigned right_rotate(unsigned x, int n)
{
```

```
int i;
int wl = sizeof(unsigned)*8;

/* Postupak se ponavlja n puta */
for (i = 0; i < n; i++)
{
    /* Poslednji bit broja x */
    unsigned last_bit = x & 1;

    /* x pomeramo za jedno mesto u desno */
    x >>= 1;

    /* Zapamceni poslednji bit stavljamo na pocetak broja x*/

    x |= last_bit<<wl-1;
}

return x;
}

/* Funkcija obrce binarni zapis neoznacenog broja x tako sto bitove
   cita unatrag */
unsigned mirror(unsigned x)
{
int i;
int wl = sizeof(unsigned)*8;

/* Rezultat inicijalizujemo na poslednji bit broja x */
unsigned y = x & 1;

/* Postupak se ponavlja wl-1 puta */
for (i = 1; i<wl; i++)
{
    /* x se pomera u desno za jedno mesto */
    x >>= 1;
    /* rezultat se pomera u levo za jedno mesto */
    y <<= 1;

    /* Poslednji bit broja x upisujemo na poslednje
       mesto rezultata */
    y |= x & 1;
}
return y;
```

```
}

main()
{
    unsigned x = 0xFAF0FAF0;
    print_bits(x);
    print_bits(mirror(x));
    print_bits(right_rotate(x, 2));
}

Izlaz iz programa:
11111010111100001111101011110000
00001111010111110000111101011111
00111110101111000011111010111100
```

1.29 Linearna i binarna pretraga

Primer 74 *Linearna pretraga*

```
#include <stdio.h>

/* Funkcija proverava da li se dati element x nalazi
   u datom nizu celih brojeva.
   Funkcija vraca poziciju u nizu na
   kojoj je x pronadjen
   odnosno -1 ukoliko elementa nema.
*/
int linearna_pretraga(int niz[], int br_elem, int x)
{
    int i;
    for (i = 0; i<br_elem; i++)
        if (niz[i] == x)
            return i;
        /* nikako else */

    return -1;
}

main()
{
    /* Inicijalizacija niza moguca je
    i na ovaj nacin*/
    int a[] = {4, 3, 2, 6, 7, 9, 11};
```

```
    /* Da bi smo odredili koliko clanova
     ima niz mozemo koristiti operator
     sizeof*/
    int br_elem = sizeof(a)/sizeof(int);
    int x;
    int i;
    printf("Unesite broj koji trazimo : ");
    scanf("%d",&x);

    i = linearna_pretraga(a, br_elem, x);
    if (i == -1)
        printf("Element %d nije nadjen\n",x);
    else
        printf("Element %d je nadjen na
            poziciji %d\n",x, i);
}
```

Primer 75 *Binarna pretraga niza*

```
/* Binarna pretraga niza celih brojeva - iterativna verzija*/
#include <stdio.h>

/* Funkcija proverava da li se element x javlja unutar niza
   celih brojeva a.
   Funkcija vraca poziciju na kojoj je element nadjen odnosno
   -1 ako ga nema.

   !!!!! VAZNO !!!!!
   Pretpostavka je da je niz a uredjen po velicini
*/

int binarna_pretraga(int a[], int n, int x)
{
    /* Pretrazujemo interval [l, d] */
    int l = 0;
    int d = n-1;

    /* Sve dok interval [l, d] nije prazan */
    while (l <= d)
    {
        /* Srednja pozicija intervala [l, d] */
        int s = (l+d)/2;
```

```
        /* Ispitujemo odnos x i a[srednjeg elementa] */
        if (x == a[s])
            /* Element je pronadjen */
            return s;
        else if (x < a[s])
        {
            /* Pretrazujemo interval [l, s-1] */
            d = s-1;
        }
        else
        {
            /* Pretrazujemo interval [s+1, d] */
            l = s+1;
        }
    }

    /* Element je nadjen */
    return -1;
}

main()
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;

    printf("Unesi element kojega trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, sizeof(a)/sizeof(int), x);

    if (i==-1)
        printf("Elementa %d nema\n", x);
    else
        printf("Pronadjen na poziciji %d\n", i);
}
```

1.30 Razni zadaci

Primer 76 *Obrtanje stringa i pretvaranje broja u string*

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

```
/* reverse:  obrce string, npr string "1234" postaje "4321" */
void reverse(char s[])
{
    int c, i, j;

    for (i = 0, j = strlen(s)-1; i < j; i++, j--)
    {
        c = s[i];
        s[i] = s[j];
        s[j] = c;
    }
}

/* itoa:  konvertuje broj n u niz karaktera s */
void itoa(int n, char s[])
{
    int i, sign;

    if ((sign = n) < 0)  /* sacuvaj znak */
        n = -n;          /* napravi da je n pozitivno */
    i = 0;
    do {
    /* generisanje cifara u obrnutom smeru */
    s[i++] = n % 10 + '0';  /* izracunaj sledecu cifru */
    } while ((n /= 10) > 0);    /* izbaci cifru iz zapisa */
    if (sign < 0)
        s[i++] = '-';
    s[i] = '\0';
    reverse(s);
}

main()
{
int n=-44;
char nst[100];
itoa(n,nst);
printf("%s\n",nst);
}
```

Primer 77 *btoi - konverzija iz datog brojnog sistema u dekadni.*

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

```
/* Pomocna funkcija koja izracunava vrednost
    koju predstavlja karakter u datoj osnovi
   Funkcija vraca -1 ukoliko cifra nije validna.

   Npr.
   cifra 'B' u osnovi 16 ima vrednost 11
   cifra '8' nije validna u osnovi 6
*/

int digit_value(char c, int base)
{
/* Proveravamo obicne cifre */
if (isdigit(c) && c < '0'+base)
    return c-'0';

/* Proveravamo slovne cifre za mala slova */
if ('a'<=c && c < 'a'+base-10)
    return c-'a'+10;

/* Proveravamo slovne cifre za velika slova */
if ('A'<=c && c < 'A'+base-10)
    return c-'A'+10;

return -1;
}

/* Funkcija izracunava vrednost celog broja koji je zapisan u datom
   nizu karaktera u datoj osnovi. Za izracunavanje se koristi
   Hornerova shema.
*/
int btoi(char s[], int base)
{
int sum = 0;

/* Obradjuju se karakteri sve dok su to validne cifre */
int i, vr;
for (i = 0; (vr = digit_value(s[i], base)) != -1; i++)
    sum = base*sum + vr;

return sum;
}

main()
```

```
{
char bin[] = "11110000";
char hex[] = "FF";

printf("Dekadna vrednost binarnog broja %s je %d\n", bin, btoi(bin, 2));
printf("Dekadna vrednost heksadekadnog
                                broja %s je %d\n", hex, btoi(hex, 16));
}
```

Primer 78 *Učitava linije sa ulaza i pamti najdužu liniju.*

```
#include <stdio.h>
#define MAXLINE 1000 /* maximum input line length */

int getline(char line[], int maxline);
void copy(char to[], char from[]);

/* print the longest input line */
main()
{
    int len;   /* current line length */
    int max;   /* maximum length seen so far */
    char line[MAXLINE]; /* current input line */
    char longest[MAXLINE]; /* longest line saved here */

    max = 0;
    while ((len = getline(line, MAXLINE)) > 0)
        if (len > max) {
            max = len;
            copy(longest, line);
        }
    if (max > 0)  /* there was a line */
        printf("%s", longest);
}

/* getline:  read a line into s, return length  */
int getline(char s[],int lim)
{
    int c, i;

    for (i=0; i < lim-1
            && (c=getchar())!=EOF && c!='\n';++i)
        s[i] = c;
    if (c == '\n') {
```

```
        s[i] = c;
        ++i;
    }
    s[i] = '\0';
    return i;
}

/* copy:  copy 'from' into 'to';
assume to is big enough */
void copy(char to[], char from[])
{
    int i;

    i = 0;
    while ((to[i] = from[i]) != '\0')
        ++i;
}
```

Primer 79 *Konvertovanje stringa u broj u pokretnom zarezu.*

```
#include <ctype.h>
#include <stdio.h>
#define MAXLINE 100

/* getline:  get line into s, return length */
int getline(char s[], int lim)
{
int c, i;

i = 0;
while (--lim > 0 && (c=getchar()) != EOF && c != '\n')
    s[i++] = c;
if (c == '\n')
    s[i++] = c;
s[i] = '\0';
return i;
}

/* atof:  convert string s to double */
double atof(char s[])
{
double val, power;
int i, sign;
```

```
/* skip white space */
for (i = 0; isspace(s[i]); i++)
    ;
/* Postavlja se znak broja*/
sign = (s[i] == '-') ? -1 : 1;

/* Preskace se jedno mesto ukoliko je bio upisan znak u broj*/
if (s[i] == '+' || s[i] == '-')
    i++;

/* Racuna se vrednost broja dok se ne naidje na tacku */
for (val = 0.0; isdigit(s[i]); i++)
    val = 10.0 * val + (s[i] - '0');

if (s[i] == '.')
    i++;

/* Racuna se vrednost broja iza tacke*/
for (power = 1.0; isdigit(s[i]); i++)
    {
    val = 10.0 * val + (s[i] - '0');
    power *= 10;
    }
return sign * val / power;
}

/* Program sabira brojeve u pokretnom zarezu koji se unose sa ulaza*/
main()
{
double sum;
char line[MAXLINE];

sum = 0;
while (getline(line, MAXLINE) > 0)
    printf("\t%g\n", sum += atof(line));
}
```

Primer 80 *Funkcija koja uklanja znak* `c` *kad god se pojavi u stringu* `s`*.*

```
#include <stdio.h>

void squeeze(char s[], char c)
```

```
{
int i,j;
for(i=j=0; s[i]!='\0';i++)
    if(s[i]!=c) s[j++]=s[i];
s[j]='\0';
}

main()
{
    char niz[20];
    char c;

    printf("Unesi karakter\n\n");
    scanf("%c", &c);

    scanf("%s", niz);
    squeeze(niz, c);
    printf("%s\n", niz);
}
```