Programiranje 1

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

January 28, 2007

Sadržaj

1

Programski jezik C

1.1 Tipovi, operatori i izrazi. Kontrola toka.

[1]

Primer 1 *Program na standardni izlaz štampa "Zdravo, svete!".*

```
#include <stdio.h>

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

Izlaz iz programa:
Zdravo, svete!
```

Primer 2 *Šta je izlaz iz sledećeg programa?*

```
#include <stdio.h>

   main()
   {
     printf("Zdravo, ");
     printf("svete!");
     printf("\n");
   }

Izlaz iz programa:
Zdravo, svete!
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`, `http://www.matf.bg.ac.yu/~milena`.

1.2 Imena promenljivih

Postoje ograničenja: u imenu se mogu pojaviti slova i cifre, potcrta "_" se smatra slovom (uglavnom se koristi kod dužih imena promenljivih).
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 Aritmetičke i relacijske operacije

Primer 7 *Program vrši oduzimanje dva cela broja.*

```
#include <stdio.h>

main()
 {
 /*deklaracija vise promenljivih istog tipa */
int rez,pom1,pom2; /*rezultat oduzimanja pom1-pom2 -> rez*/
 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
```

Primer 8 *Program sabira dva uneta cela broja*

```
#include <stdio.h>

int 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);
    return 0;
}
Ulaz:
Unesi prvi broj : 2 <enter>
Unesi drugi broj : 3 <enter>
Izlaz:
2 + 3 = 5
```

Primer 9 *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);
```

```
        printf("Zbir a+b je : %d\n",a+b);
        printf("Razlika a-b je : %d\n",a-b);
        printf("Proizvod a*b je : %d\n",a*b);
        printf("Celobrojni kolicnik a/b je : %d\n", a/b);
        printf("Pogresan pokusaj racunanja realnog kolicnika a/b je : %f\n", a/b);
        printf("Realni kolicnik a/b je : %f\n", (float)a/(float)b);
        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 10 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 3.000000 */
    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 promenjive mora kastovanje : %f\n", (float)a/(float)b);

}

Izlaz iz programa:
c = 2.000000
Uzrok problema : 5/2 = 0.000000
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
```

1.6.1 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.

1.6.2 Inkrementacija i dekrementacija

Operatori ++ i --

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

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

Primer 11 *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 promenjivu, i rezultat
       je uvecana vrednost */
    x = ++a;

    /* Postfiksni operator uvecava promenjivu, 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
```

1.6.3 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 12 *Ilustracija logičkih vrednosti (0 - netačno, razlicito od 0 - tačno).*

```
#include <stdio.h>

main()
{
    int a;

    printf("Unesi ceo broj : ");
```

```
    scanf("%d", &a);
    if (a)
        printf("Logicka vrednost broja je : tacno\n");
    else
        printf("Logicka vrednost broja je : netacno\n");
}

Ulaz:
Unesi ceo broj : 3 <enter>
Izlaz:
Logicka vrednost broja je : tacno

Ulaz:
Unesi ceo broj : 0 <enter>
Izlaz:
Logicka vrednost broja je : netacno
```

Primer 13 *Ilustracija logičkih i relacijskih operatora.*

```
#include <stdio.h>

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

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

    printf("Konjunkcija : 3>5 && 5>3 - %d\n", a && b);
    printf("Disjunkcija : 3>5 || 5>3 - %d\n", a || b);
    printf("Negacija : !(3>5) - %d\n", !a);

}

Izlaz iz programa:
5<3 - 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
```

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

1.7.1 if

```
if (izraz)
```

```
    iskaz1
else
    iskaz2
```

Primer 14 *Program ilustruje if i ispisuje ukoliko je uneti ceo broj negativan.*

```
#include <stdio.h>

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

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

Ulaz:
Unesi ceo broj:5
Izlaz:
```

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!!!

1.7.2 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 15 *Program ilustruje if-else konstrukciju i ispituje znak broja.*

```
#include <stdio.h>

int 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");
    return 0;
}

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

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

Primer 16 *Pogresan program sa dodelom = umesto poredjenja ==.*

```
#include <stdio.h>

int 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");
    return 0;
}

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

1.7.3 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;
```

1.7.4 do-while

Ovo je slično paskalskom repeat-until izrazu.

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

Primer 17 *Program ilustruje petlju - while.*

```
#include <stdio.h>

int main()
{
    int x;

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

}

Izlaz:
```

```
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
```

Primer 18 *Program ilustruje petlju do-while.*

```
#include <stdio.h>

int main()
{
    int x;

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

}

Izlaz:
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
x = 10
```

Primer 19 *Program ilustruje petlju - for.*

```
#include <stdio.h>

int main()
{
    int x;

    for (x = 1; x < 10; x++)
        printf("x = %d\n",x);
```

```
}

Izlaz:
x = 1
x = 2
x = 3
x = 4
x = 5
x = 6
x = 7
x = 8
x = 9
```

Primer 20 *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

int 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; */
    }
    return 0;
}

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

Primer 21 *Konverzija centimetara u inče - for petlja.*

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

int main()
{
```

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

    return 0;
}

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

Zadaci za vežbu:

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);
}
```

Zadatak 2 *Napisati program koji sabira dva cela broja sa ulaza.*

Zadatak 3 *Napisati program za razmenu vrednosti dva cela broja.*

Zadatak 4 *Izvršiti štampanje parnih brojeva od 1 do 100 (for, while i do-while).*

Zadatak 5 *Napisati program koji izračunava sumu i maksimum brojeva koji se unose na standardni ulaz pri čemu je poslednji uneti broj 0 (for, while).*

Zadatak 6 *Napisati program koji ispisuje kvadrate svih brojeva od 5 do 35. Nakon svakog petog kvadrata odštampati znak za novi red(for, while).*

2

Programski jezik C

[1]

2.1 Priprema za kolokvijum

Primer 22 *Šta će biti izlaz iz sledećeg programa?*

```
#include <stdio.h>

main()
{
    printf("\"Zdravo, svima\"\n");
    printf("\\n\tprelazak u novi red\n");
    printf("\\t\ttabulator\n");
    printf("\\\\\tkosa crta\n");
    printf("%%%%\tprocenat\n");
}

Izlaz iz programa:
"Zdravo, svima"
\n  prelazak u novi red
\t  tabulator
\\  kosa crta
%%  procenat
```

Primer 23 *A šta iz ovog?*

```
#include <stdio.h>

main()
{
    putchar('\\');
    putchar('t');
    putchar('\t');
    printf("Za %d ispisujem %c", '\\', '\\');
    printf("\n\n\\n\\\n\\\\n\n");
}
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`, `http://www.matf.bg.ac.yu/~milena`.

```
Izlaz iz programa:
\t  Za 92 ispisujem \

\n\
\\n
```

Primer 24

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

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

Primer 25 *Program ispisuje ascii tabelu.*

```
#include <stdio.h>

main()
{
    int c;
    for (c = 0; c<128; c++)
        printf("%d - %c\n",c,c);
}
```

Primer 26 *Napisati program za razmenu vrednosti dva cela broja (1.način).*

```
#include <stdio.h>

main()
{
    int a = 10;
    int b = 15;
    int tmp;

    tmp  = a;
    a = b;
    b = tmp;
```

```
    printf ("a=%d, b=%d\n", a, b);
}
```

Primer 27 *Napisati program za razmenu vrednosti dva cela broja (2.način).*

```
#include <stdio.h>

main()
{
    int a = 10;
    int b = 15;

    b = a+b;  /* a = 10; b = 25; */
    a = b-a;  /* a = 15; b = 25; */
    b = b-a;  /* a = 15; b = 10; */

    printf ("a=%d, b=%d\n", a, b);
}
```

Primer 28 *Program menja mesta cifara u broju.*

```
#include <stdio.h>
main(){
  int n,t=0;
  printf("Unesite broj\n");
  scanf("%d",&n);
   while(n)
    { t=t*10+n%10;
      n/=10;
     }

  printf("Novi broj je %d\n", t);
  return 0;
}

Izlaz:
Unesite broj
1234
Novi broj je 4321
```

Zadaci za vežbu:

Zadatak 7 *Dat je fragment C programa:*

```
i=1; j=1;
while (i+j<10)
 { ++j; i+=2;}
suma=i+j;
```

a) Koliko puta će se ponoviti while ciklus?
b) Koje su vrednosti promenljivih i, j, suma nakon izvršenja fragmenta?
c) Napisati ekvivalentan for ciklus.

Zadatak 8 *Dat je fragment C programa:*

```
i=1; j=1;
while (i+j<10)
  ++j; i+=2;
suma=i+j;
```

a) Koliko puta će se ponoviti while ciklus?
b) Koje su vrednosti promenljivih i, j, suma nakon izvršenja fragmenta?
c) Napisati ekvivalentan for ciklus.

Zadatak 9 *Napisati program koji izračunava zbir recipročnih vrednosti prvih 10 brojeva.*

Zadatak 10 *Napisati program koji izračunava maksimum i minimum 3 cela broja sa ulaza.*

3

Programski jezik C

[1]

3.1 Switch

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

Primer 29 *Ilustracija switch konstrukcije.*

```
#include<stdio.h>

int 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");
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`,`http://www.matf.bg.ac.yu/~milena`.

```
            break;
        defalut:
            printf("Uneli ste nesto sto nije paran broj\n");
    }
    return 0;
}
```

```
Ulaz: Unesi paran broj manji od 10 2 Izlaz: Uneli ste dvojku
```

3.2 Uslovni izraz

```
izraz1 ? izraz2 : izraz3

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

max = (a>b)? a : b;
```

3.3 Operator sizeof

Primer 30 *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
```

3.4 Znakovni ulaz i izlaz - getchar i putchar

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 31 *Program vrši demonstraciju funkcija putchar i getchar.*

```
#include <stdio.h>

main()
{
    int  c1, c2;
    c1 = getchar();
    printf("------------\n");
    c2 = getchar();

    printf("c1 = %d, c2 = %d\n",c1, c2);
    printf("c1 = %c, c2 = %c\n",c1, c2);

    putchar(c1); /* isto je kao i printf("%c",c1); */
    putchar(c2); /* isto je kao i printf("%c",c2); */
    putchar('\n');

    /* Za ispisivanje karaktera a */
    putchar('a');
    /* dozvoljeno je :  printf("abc"); printf("a"); */
    /* nedozvoljeno je : printf('a'); putchar('abc'); putchar("abc"); */
}

Ulaz: ab

Izlaz:
------------
c1 = 97, c2 = 98 c1 = a, c2 = b ab a
```

Primer 32 *Program čita jedan karakter i ispisuje ga.*

```
#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 33 *Program vrši prebrojavanje cifara unetih na ulazu.*

```
#include <stdio.h>

/* zbog isdigit */
#include <ctype.h>
main()
{
    int c;
    int br_cifara = 0;
    while ((c = getchar()) != EOF)
        if ('0'<=c && c<='9') /* moze i if (isdigit(c)) */
            br_cifara++;

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

Primer 34 *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 35 *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 36 *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 37 *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 38 *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: niti case '0'; */
            case '0':
                br_0++;
                break;  /* Isprobati veziju bez break */
            case '1':
                br_1++;
                break;
            case '2':
                br_2++;
                break;
        }
    }
    printf("Br 0 : %d\nBr 1 : %d\nBr 2 : %d\n",br_0, br_1, br_2);
}
```

3.5 Ugnježdena petlja

Primer 39 *Ilustracija dve ugnježdene petlje.*

```
#include<stdio.h>

int main()
{
    int i,j;
    for(i=1; i<=3; i++)
      {
         for(j=1; j<=3; j++)
         printf("%d * %d = %d\t", i, j, i*j);
         printf("\n");
     }
}

Izlaz:
1 * 1 = 1       1 * 2 = 2       1 * 3 = 3
```

```
2 * 1 = 2       2 * 2 = 4       2 * 3 = 6
3 * 1 = 3       3 * 2 = 6       3 * 3 = 9
```

Primer 40 *Program koji ispisuje tablicu množenja*

```
#include<stdio.h>

main() {
 int n, m;  /* Dimenzije tablice */
 int i, j;  /* Brojaci */

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

 /* Petlja po redovima... */
 for(i = 0; i < n; i++) {
    /* unutrasnja petlja */
    for(j = 0; j < m; j++)
        printf("%d * %d = %d\t", i, j, i*j);
    /* na kraju prelazimo u sledeci red */
    printf("\n");
 }
}
```

Primer 41 *Program koji ispisuje prvih n prostih brojeva*

```
#include<stdio.h>

main() {
 int i, n, br, delilac, ostatak;

 printf("Unesite koliko prostih brojeva zelite da dobijete: \n");
 scanf("%d", &n);

 /* Inicijalizujemo brojac i - koliko smo prostih brojeva nasli do sad */
 i = 0;

 /* Pocetni broj za koji proveravamo da li je prost */
 br = 2;

 /* Trazimo i-ti prost broj */
 while(i < n) {
    /* Ako je u pitanju 2 ili 3 prost je */
    if (br <= 3)
        p = 1;
    else if (br % 2 == 0)
        /* ako je broj paran i veci od 2 onda nije prost */
        p = 0;
    else {
        /* Ispitujemo samo neparne pa delioci mogu biti samo neparni
        brojevi */
        delilac = 3;
        ostatak = 1;
```

```
        while(ostatak != 0 && delilac * delilac <= n) {
            ostatak = n % delilac;
            delilac++;
        }
        p = (ostatak != 0);
    }

    /* Ako je broj prost... */
    if (p) {
        /* stampamo ga... */
        printf("Broj %d je prost.\n", n);
        /* i uvecavamo broj pronadjenih prostih brojeva. */
        i++;
     }

 /* U svakom slucaju prelazimo na proveru da li je sledeci broj prost */
 br++;
 }
}
```

3.6 Oblast važenja lokalnih promenljivih

Primer 42

```
#include <stdio.h>

main()
{
    int pom=1;
    printf("Pre ulaska u unutrasnji blok pom=%d\n",pom);
    {
        int pom=50;
        printf("Pre izlaska iz unutrasnjeg bloka  pom=%d\n",pom);
    }
    printf("Nakon izlaska iz  unutrasnjeg bloka  pom=%d\n",pom);
}
Izlaz: Pre ulaska u unutrasnji blok pom=1
Pre izlaska iz unutrasnjeg bloka pom=50
Nakon izlaska iz  unutrasnjeg bloka pom=1
```

Zadaci za praktikum:

Zadatak 11 *Napisati program koji broji linije i znakove sa ulaza.*

Zadatak 12 *Napisati program koji prepisuje ulaz na izlaz čineći tabulatore, nove linije i backslash-ove vidljivim.*

Zadatak 13 *Sledeći program koji prepisuje standardni ulaz na standardni izlaz pokrenuti sa: (ilustracija redirekcije standardnog ulaza i izlaza)*

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

```
#include <stdio.h>

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

Zadatak 14 *Napisati program koji pronalazi maksimum brojeva sa ulaza - verzija bez niza.*

Zadatak 15 *Napisati program koji sabira pozitivne brojeve niza cifara koji završava nulom i koji se unose sa standardnog ulaza.*

Primer 43 *Napisati program koji računa zbir* $1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$

Primer 44 *Napisati program koji računa sumu* $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n * \frac{x^{2n-1}}{(2n-1)!}$

Primer 45 *Napisati program koji računa sumu* $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$

Primer 46 *Napisati program koji računa sumu* $x - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)*n!}$

4

Programski jezik C

[1]

4.1 Nizovi — osnovni pojmovi

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!
Indeks niza može da bude proizvoljan izraz celobrojne vrednosti: `niz[i*2]=5`.

Primer 47 *Program ilustruje korišćenje statičkih nizova. Ispisuje 10 unetih brojeva unazad.*

```
#include <stdio.h>

main()
{
    int a[10];
    int i;
    for (i = 0; i<10; i++)
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`,`http://www.matf.bg.ac.yu/~milena`.

```
    {   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 48 *Brojanje pojavljivanja svake od cifara. Koriscenje niza brojača.*

```
#include <stdio.h>
#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 49 *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]);

}
```

4.2 Funkcije

Primer 50 *sum - najjednostavnija funkcija koja sabira dva broja*

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

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

Primer 51 *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.*

```
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 52 *power - funkcija koja stepenuje realan broj na celobrojni izlozilac*

```
 #include <stdio.h>

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

    return s;
}

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

Primer 53 *Verzija koja radi i za negativne izlozioce*

```
int power_n(float x, int k)
{
    int i;
    int negative = k<0;

    if (negative)
        k = -k;

    float s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return negative ? 1.0/s : s;
}

main()
{
    /* Poziv funkcije */
    float s = power(2.0,-1);
    printf("%f\n", s);
}
```

Primer 54 *Napisati funkciju koja izračunava zbir n-tih stepena brojeva od 1 do granice i program koji ilustruje rad ove funkcije.*

```
#include <stdio.h>
void Zbir_stepena (int n, int granica);
main()
{
```

```
    Zbir_stepena(2,5);
    Zbir_stepena(3,5);
    Zbir_stepena(4,10);
    return 0;
}

void Zbir_stepena (int n, int granica)
{

    int i,j;   /*brojaci u for petljama */
    long Zbir=0  ,  stepenovan ;

     /*spoljasnji for ciklus obavlja sumiranja*/
    for (i=1; i<=granica; Zbir +=stepenovan, ++i)
        /*unutrasnji for ciklus obavlja stepenovanje */
         for( stepenovan=1,j=1;  j<=n;  stepenovan*= (long) i, ++j) ;
    printf(" Zbir %d. stepena od 1 do %d jeste %ld\n", n,granica,Zbir);
}

Izlaz:
 Zbir 2. stepena od 1 do 5 jeste 55
 Zbir 3. stepena od 1 do 5 jeste 225
 Zbir 4. stepena od 1 do 10 jeste 25333
```

Primer 55 *Napisati funkciju koja izračunava zbir kvadrata brojeva od 1 do date granice kao i program koji ilustruje korišćenje date funkcije.*

```
#include <stdio.h>
void  Zbir_Kvad(int n);   /*f-ja koja vrsi zeljeno izracunavanje */
main()
{
    Zbir_Kvad( 5);
    Zbir_Kvad( 23);
}
void  Zbir_Kvad(int n)
{
    int br; /* lokalna promenljiva funkcije, brojac u ciklusu */
    long Zbir=0; /* lokalna promenljiva funkcije, suma kvadrata brojeva od 1..n */
    for  (br=1; br<=n;  Zbir+= (long)  br*br,  ++br)  ;
    printf(" Zbir kvadrata brojeva od 1 do %d jese %ld\n", n,Zbir);
}

Izlaz:
Zbir kvadrata brojeva od 1 do 5 jese 55
Zbir kvadrata brojeva od 1 do 23 jese 4324
```

Primer 56 *Napisati program u C-u koji prikazuje sve proste brojeve u datom intervalu kojima je zbir cifara složen broj. Interval se zadaje učitavanjem gornje i donje granice (dva prirodna broja). Brojeve prikazati u opadajućem poretku.*

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

```
int prost (int n);  /*testira da li je broj n prost broj */
/*Prirodni brojevi (sem 1)imaju najmanje dva delioca:jedinicu i samog sebe.
Brojevi koji nemaju drugih delioca,sem ova dva, nazivaju se prostim */

int zbirCifara (int n);   /*vraca zbir cifara broja n */
main()
{
    int donja,gornja;  /*granice intervala */
    int i;  /*brojac u petlji */
    int pom;   /*posrednik u eventualnoj zameni */

  /*ucitavanja granice intervala */
    scanf("%d%d", &donja, &gornja);
     if (donja > gornja)   /*obezbedjivanje relacije: donja <=gornja */
    {
        pom=donja;
        donja=gornja;
         gornja=pom;
     }
     for(i=gornja;i>=donja; i--)
     if (prost (i) && !prost(zbirCifara(i) )  ) printf("%d\n",i);
}

int prost(int n)
/*Ispituje se da li je broj n prost tako to se proverava da li ima delioce
medju brojevima od 2 do n/2.  Pri implementaciji se koristi tvrdjenje da je
broj prost ako je jednak 2, ili ako je neparan i ako nema delitelja medju
neparnim brojevima od 3 do n/2 */
{
    int prost;  /*indikator slozenosti broja n */
    int i;    /*potencijalni delitelj broja n */
    if (n==1) return 0;
     /*parni brojevi razliciti od  od dva nisu prosti brojevi */
     prost= (n%2!=0) || (n==2);

     /*najmanji potencijalni kandidat za delitelje medju
      neparnim brojevima razlicitim od jedan */
     i=3;
    while ( (prost) && (i<=n/2) )
   {
    prost=n%i != 0;
    i=i+2; /*proveravamo kandidate za delitelje samo medju neparnim brojevma */
   }
 return prost;
}
int zbirCifara (int n)
{ int  Suma=0;
   while (n>0)
  {
  Suma+= n%10;  /*dodavanje cifre tekuceg razreda,pocev od razreda jedinica ,
                  a iduci ka visim razredima cifara */
```

```
  n=n/10;      /*prelaz ka visem razredu */
  }
 return Suma;
}
```

```
Ulaz:
1 20
Izlaz:
19
17
13
```

Zadaci za vežbu:

Zadatak 16 *Sa tastature učitati elemente niza koji su celi brojevi i za koje se pretpostavlja da ih nema više od 100. Pronaći maksimalan elemenat niza i ispisati ga na izlaz.*

Zadatak 17 *Sa tastature se unosi 15 karaktera u niz. Ispitati da li uneti niz predstavlja palindrom (primer palindroma je "anavolimilovana").*

Zadatak 18 *Ispisati prvih 15 članova Fibonačijevog niza.*

Zadatak 19 *Napisati program koji ispituje da li dva niza imaju barem jedan zajednički element.*

Zadatak 20 *Napisati f-ju koja za uneti broj n izračunava zbir recipročnih vrednosti prvih n brojeva.*

Zadatak 21 *Ilustracija korišćenja funkcije za izračunavanje faktorijela celog broja.*

(a) Napisati program koji izračunava faktorijel unetog broja.

(b) Napisati funkciju koja izračunava faktorijel celog broja.

(c) Napisati program koji izračunava faktorijel unetog broja koristeći prethodno definisanu funkciju.

Zadatak 22 *Ilustracija korišćenja funkcije za proveru da li je broj prost.*

(a) Napisati program koji za uneti broj proverava da li je prost.

(b) Napisati funkciju koja za ceo broj proverava da li je prost.

(c) Napisati program koji štampa prvih 100 prostih brojeva.

5

Programski jezik C

[1]

5.1 Pokazivači

Pokazivač je promenljiva koja sadrži adresu promenljive.

```
int x=1, y=1, z[10];
int *ip;    /* ip je pokazivac na int,
               odnosno *ip je tipa int*/

ip = &x;    /* ip sada pokazuje na x */
y=*ip;      /* y je sada 1 */
*ip = 0;    /* x je sada 0 */

*ip+=10;    /* x je sada 10*/
++*ip;      /* x je sada 11*/
(*ip)++;    /* x je sada 12,
            zagrada neophodna zbog prioriteta
            operatora*/

ip = &z[0]; /* ip sada pokazuje na z[0]*/
```

Primer 57 *Ilustracija rada sa pokazivačkim promenljivim.*

```
#include <stdio.h>
main() {
    int x = 3;

    /* Adresu promenjive x zapamticemo u novoj promeljivoj.
       Nova promenljiva je tipa pokazivaca na int (int*) */
    int* px;

    printf("Adresa promenljive x je : %p\n", &x);
    printf("Vrednost promenljive x je : %d\n", x);
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`,`http://www.matf.bg.ac.yu/~milena`.

```
    px = &x;
    printf("Vrednost promenljive px je (tj. px) : %p\n", px);
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Menjamo vrednost promenljive na koju ukazuje px */
    *px = 6;
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Posto px sadrzi adresu promenljive x, ona ukazuje na x tako da je
       posredno promenjena i vrednost promenljive x */
    printf("Vrednost promenljive x je : %d\n", x);


}

Izlaz (u konkretnom slucaju):
Adresa promenljive x je : 0012FF88
Vrednost promenljive x je : 3
Vrednost promenljive px je (tj. px) : 0012FF88
Vrednost promenljive na koju ukazuje px (tj. *px) je : 3
Vrednost promenljive na koju ukazuje px (tj. *px) je : 6
Vrednost promenljive x je : 6
```

Pored pokazivača na osnovne tipove, postoji i pokazivač na prazan tip (void).

```
void *pp;
```

Njemu može da se dodeli da pokazuje na int, ili na char ili na proizvoljan tip ali je to neophodno eksplicitno naglasiti svaki put kada želimo da koristimo ono na šta on pokazuje.

Primer 58 *Upotreba pokazivača na prazan tip.*

```
#include<stdio.h>

main()
{
void *pp;
int x=2;
char c='a';

pp = &x;
*(int *)pp = 17;    /* x postaje 17*/
printf("\n adresa od x je %p", &x);
printf("\n%d i %p",*(int*)pp,(int * )pp);

pp = &c;
printf("\n adresa od c je %p", &c);
printf("\n%c i %p",*(char*)pp,(char * )pp);

}

/*
 adresa od x je 0012FF78
17 i 0012FF78
```

```
 adresa od c je 0012FF74
a i 0012FF74

*/
```

Posebna konstanta koja se koristi da se označi da pokazivač ne pokazuje na neko mesto u memoriji je NULL.

5.2 Prenos parametara po vrednosti i preko pokazivača

Primer 59 *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);
}
Izlaz:
3
```

C prosleđuje argumente u funkcije pomoću vrednosti. To znači da sledeća funkcija neće uraditi ono što želimo:

```
void swap (int x, int y) /* POGRESNO!!!!!!!!*/
{
int temp;
temp = x;
x=y;
y=temp;
}
```

Zbog prenosa parametara preko vrednosti swap ne može da utiče na argumente a i b u funkciji koja je pozvala swap. Ova swap funkcija samo zamenjuje kopije od a i b.

Da bi se dobio željeni efekat, potrebno je da se proslede pokazivači:

```
/* Zameni *px i *py */
void swap (int *px, int *py)
{
int temp;
temp =*px;
*px = *py;
*py = temp;
}
```

a poziv funkcije swap izlgeda sada ovako

```
swap(&a, &b);
```

Primer 60 *Demonstracija više povratnih vrednosti funkcije koristeći prenos preko pokazivača.*

```
/* Funkcija istovremeno vraca dve vrednosti - kolicnik i ostatak
dva data broja.
   Ovo se postize tako sto se funkciji predaju vrednosti dva broja (x i y) koji se dele
   i adrese dve promenljive na koje ce se smestiti rezultati */
void div_and_mod(int x, int y, int* div, int* mod) {
    printf("Kolicnik postavljam na adresu : %p\n", div);
    printf("Ostatak postavljam na adresu : %p\n", mod);
    *div = x / y;
    *mod = x % y;
}

main() {
    int div, mod;
    printf("Adresa promenljive div je %p\n", &div);
    printf("Adresa promenljive mod je %p\n", &mod);

    /* Pozivamo funkciju tako sto joj saljemo vrednosti dva broja (5 i 2)
       i adrese promenljvih div i mod na koje ce se postaviti rezultati */
    div_and_mod(5, 2, &div, &mod);

    printf("Vrednost promenljive div je %d\n", div);
    printf("Vrednost promenljive mod je %d\n", mod);

}

Izlaz u konkretnom slucaju:
Adresa promenljive div je 0012FF88
Adresa promenljive mod je 0012FF84
Kolicnik postavljam na adresu : 0012FF88
Ostatak postavljam na adresu : 0012FF84
Vrednost promenljive div je 2
Vrednost promenljive mod je 1
```

5.3 Lenjo izračunavanje

Primer 61 *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>

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 se pozivati 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);
}

Izlaz:
Pozvano izracunaj()
Uslov ispunjen : a = 0, b = 1
Uslov nije ispunjen : a = 1, b = 1
Pozvano izracunaj()
Uslov ispunjen : a = 2, b = 2
Uslov nije ispunjen : a = 3, b = 2
Pozvano izracunaj()
Uslov ispunjen : a = 4, b = 3
Uslov nije ispunjen : a = 5, b = 3
Pozvano izracunaj()
Uslov ispunjen : a = 6, b = 4
Uslov nije ispunjen : a = 7, b = 4
Pozvano izracunaj()
Uslov ispunjen : a = 8, b = 5
Uslov nije ispunjen : a = 9, b = 5
----------------------------
Uslov ispunjen : a = 0, b = 0
Pozvano izracunaj()
Uslov ispunjen : a = 1, b = 1
Uslov ispunjen : a = 2, b = 1
Pozvano izracunaj()
Uslov ispunjen : a = 3, b = 2
Uslov ispunjen : a = 4, b = 2
Pozvano izracunaj()
Uslov ispunjen : a = 5, b = 3
```

```
Uslov ispunjen : a = 6, b = 3
Pozvano izracunaj()
Uslov ispunjen : a = 7, b = 4
Uslov ispunjen : a = 8, b = 4
Pozvano izracunaj()
Uslov ispunjen : a = 9, b = 5
```

5.4 Zadaci za vežbu

Zadatak 23 *Sledeći fragment programa obilazi istovremeno nizove sve dok nije a[i]=b[i]=0 povećavajući b[i] za 1 svaki put. Da li je program korektan? obrazložiti.*

```
int a[10], b[10];
int i = 0;
...
while(a[i]||b[i]++)i++;
```

Zadatak 24 *Sledeći deo programa obilazi niz t sleva i sdesna istovremeno i zaustavlja se kada su t[i] i t[j] različiti od 0. Da li je program korektan? obrazložiti.*

```
int t[100];
int i=0,j=100;
while(!t[i++] || !!t[--j]);
```

Zadatak 25 *Broj je Armstrongov ako je jednak sumi n-tih stepena svojih cifara. Ispitati da li je broj koji se unosi sa standardnog ulaza Armstrongov.*

Zadatak 26 *Za dati broj može se formirati niz tako da je svaki sledeći član niza dobijen kao suma cifara prethodnog člana niza. Broj je srećan ako se dati niz završava sa jedinicom. Napisati program koji za uneti broj određuje da li je srećan.*

Zadatak 27 *Sa ulaza se unosi broj u osnovi deset i osnova* $<= 10$. *Odštampati vrednost datog broja u datoj osnovi.*

Zadatak 28 *Sa ulaza se unosi osnova* $<= 10$ *i broj. Proveriti da li je taj broj ispravan broj za datu osnovu i ako jeste izračunati njegovu vrednost u osnovi 10.*

Zadatak 29 *Broj je* **Nivenov** *ako je deljiv sumom svojih cifara.*

1. *Napsati funkciju koja računa sumu cifara broja* ***a****. Na primer, za broj* `121` *funkcija treba da vrati* `4`.
2. *Napisati funkciju koja proverava da li je broj* Nivenov *i vraća* `1` *ako jeste a* `0` *ako nije.*
3. *Napisati program koji za uneto* `n` *ispisuje prvih* `n` Nivenovih *brojeva.*
4. *Napisati program koji za uneto* `n` *ispisuje sve* Nivenove *brojeve manje od* `n`.

Zadatak 30 *Napisati program koji izračunava vrednost polinoma u tački x:*

1. *Napisati funkciju koja računa k-ti stepen prirodnog broja n.*
2. *Napisati program koji za uneti niz koeficijenata* $a[i]$ *i uneti broj x računa vrednost polinoma* $a_n * x^n + a_{n-1} * x^{n-1} + ... + a_1 * x + a_0$

6

Programski jezik C

[1]

6.1 Prenos niza u f-ju

Primer 62 *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 63 *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!
```

[1] Zasnovano na primerima sa sajtova http://www.matf.bg.ac.yu/∼filip,http://www.matf.bg.ac.yu/∼milena.

```
   Nizove je neophodno prenositi zajedno sa dimenzijom niza
   (osim niski karaktera)
*/
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));
}

Izlaz:
sizeof(a) - u okviru main : 36
1 2 3 4 5 6 7 8 9
sizeof(a) - u okviru fje : 4
```

Primer 64 *Skalarni proizvod dva niza brojeva*

```
#include <stdio.h>

long mnozi(int x[],int y[],int n);

main()
{
  int a[]={1,2,3,4,5,6}, b[]={8,7,6,5,4,3};
  printf("Skalarno a*b= %ld\n",mnozi(a,b,6));
}
long mnozi(int x[ ],int y[ ],int n) { int br;
   long suma=0;
   for(br=0;br<n;br++) suma=suma+x[br]*y[br];
   return suma;
}

Izlaz:
Skalarno a*b= 98
```

6.1.1 Funkcije za rad sa stringovima

Primer 65 *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]; i++)
        putchar(s[i]);
}


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

Primer 66 *string_reverse - obrće nisku karaktera.*

```
#include <stdio.h>

/* Zbog funkcije strlen */
#include <string.h>

/* Ova funkcija racuna duzinu date niske karaktera.
   Umesto nje, moguce je koristiti standardnu funkciju strlen .
 */
int string_length(char s[])
{
    int i;
    for (i = 0; s[i]; 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 67 *Uklanja beline, tabulatore ili znak za kraj reda sa kraja stringa*

```
/* trim:  remove trailing blanks, tabs, newlines */
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 68

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

Primer 69 *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;
    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
```

Primer 70 *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);

}

Izlaz:
Unesi karakter
i
Unesi string
primer
prmer
```

Primer 71 *Dopisivanje stringova*

```
#include <stdio.h>

/* strcat:  concatenate t to
end of s; s must be big enough */
void strcat(char s[], char t[])
{
    int i, j;

    i = j = 0;
    while (s[i] != '\0') /* nadji kraj od s */
        i++;
    while ((s[i++] = t[j++]) != '\0') /* copy t */
        ;
}

main()
{
char spojeni[100]="Ovo je prvi";
char nastavak[100]=",a ovo nastavak";

printf("%s\n",spojeni);
printf("%s\n",nastavak);

strcat(spojeni,nastavak);

printf("%s\n",spojeni);
}

Izlaz:
Ovo je prvi
,a ovo nastavak
```

```
Ovo je prvi,a ovo nastavak
```

Zadaci za vežbu

Zadatak 31 *Napisati program koji u ucitanoj niski karaktera sa ulaza prebrojava pojavu cifara.*
Ilustracija redirekcije standardnog ulaza i izlaza :
pokrenuti program sa :

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

Zadatak 32 *Napisati funkciju koja vraća prvu poziciju u niski s1 na kojoj se pojavljuje znak iz s2 ili -1 ako s1 ne sadrži ni jedan znak iz s2. Ako je s1 pera a s2 navip onda funkcija treba da vrati poziciju 0. Ako je s1 zeleno a s2 nana onda funkcija treba da vrati poziciju 4.*

Zadatak 33 januar 2006.(II grupa)

1. *Napisati funkciju* ***void brojanje(int a[], int brojac[], int N)*** *čiji su argumenti* `a` *i* `brojac` *celobrojni nizovi dimenzije* `N`*. Vrednosti elemenata niza* `a` *su između* `0` *i* `N - 1`*. Funkcija izračunava elemente niza brojac tako da je* `brojac[i]` *jednak broju pojavljivanja broja* `i` *u nizu* `a`*.*

2. *Kažemo da je celobrojni niz* `a` *dimenzije* `N` *permutacija ako sadrži svako* `i: 0<=i<N`*. Sastaviti funkciju* ***int DaLiJePermutacija(int a[], int N)*** *koja vraća* `1` *ako je niz* `a` *permutacija, a inače* `0`*. (Koristiti funkciju* `brojanje`*)*

Zadatak 34 I kolokvijum, 18.januar 2006.(I grupa) *Neka je dat niz* `X` *od* `N` *nenegativnih celih brojeva. Sastaviti funkciju koja će iz niza* `X` *izbacivati sva pojavljivanja broja* `0` *i popunjavati ta mesta u nizu tako što će se preostali elementi niza pomerati ka početku niza. Odrediti i novu dimenziju* `N` *niza* `X`*. Npr. ulaz:* `N = 10, X = 0 22 11 2 0 17 33 4 0 999` *izlaz:* `N = 7, X = 22 11 2 17 33 4 999`*.*

Zadatak 35 I kolokvijum, februar 2005.

1. *Napisati funkciju koja ispituje da li dve niske (koje se prenose kao parametri funkcije) su anagrami. Anagrami su niske koje se sastoje od istih karaktera. Npr. vetar, trave, verat su anagrami.*

2. *Napisati program koji testira funkciju iz prvog dela.*

Zadatak 36 I kolokvijum, februar 2005. *Napisati program koji učitava sa standardnog ulaza dve niske sa ne više od 80 karaktera u svakoj i prirodan broj* `k` *i ispisuje na standardni izlaz poruku da li se prva niska dobila cikličnim pomeranjem druge niske za* `k` *mesta. Na primer za* `k=3`*, niska CDEAB"" se dobila cikličnim pomeranjem niske "ABCDE"*

Zadatak 37 Jun, 2004. *Napisati funkciju koja koja kao argumente prihvata dve niske i proverava da li se prva od zadatih niski može dobiti cikličnim pomeranjem karaktera druge niske.*

7

Programski jezik C

[1]

7.1 Linearna i binarna pretraga niza

Primer 72 *Linearno pretraživanje*

```
#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
    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;
```

[1] Zasnovano na primerima sa sajtova http://www.matf.bg.ac.yu/∼filip,http://www.matf.bg.ac.yu/∼milena.

```
    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 73 *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 nije nadjen */
    return -1;
}

main()
```

```
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;
    printf("Unesi element koji 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);
}
```

7.2 Životni vek i oblast važenja promenjivih. Statičke promenljive

Primer 74 *Demonstracija zivotnog veka i oblasti vazenja promenjivih (scope).*

```
#include <stdio.h>

/* Globalna promenjiva */
int a = 0;

/* Uvecava se globalna promenjiva a */
void increase()
{
    a++;
    printf("increase::a = %d\n", a);
}

/* Umanjuje se lokalna promenjiva a. Globalna promenjiva zadrzava svoju vrednost. */
void decrease()
{
    /* Ovo a je nezavisna promenjiva u odnosu na globalno a */
    int a = 0;
    a--;
    printf("decrease::a = %d\n", a);

}

void nonstatic_var()
{
    /* Nestaticke promenjive ne cuvaju vrednosti kroz pozive funkcije */
    int s=0;
    s++;
    printf("nonstatic::s=%d\n",s);
}

void static_var()
{
    /* Staticke promenjive cuvaju vrednosti kroz pozive funkcije.
       Inicijalizacija se odvija samo u okviru prvog poziva. */
    static int s=0;
```

```
    s++;
    printf("static::s=%d\n",s);
}

main()
{
    /* Promenjive lokalne za funkciju main */
    int i;
    int x = 3;

    printf("main::x = %d\n", x);


    for (i = 0; i<3; i++)
    {
        /* Promenjiva u okviru bloka je nezavisna od spoljne promenjive.
           Ovde se koristi promenjiva x lokalna za blok petlje koja ima
           vrednost 5, dok originalno x i dalje ima vrednost 3*/
        int x = 5;
        printf("for::x = %d\n", x);
    }

    /* U ovom bloku x ima vrednost 3 */
    printf("main::x = %d\n", x);

    increase();
    decrease();

    /* Globalna promenjiva a */
    printf("main::a = %d\n", a);

    /* Demonstracija nestatickih promenjivih */
    for (i = 0; i<3; i++)
        nonstatic_var();

    /* Demonstracija statickih promenjivih */
    for (i = 0; i<3; i++)
        static_var();
}

Izlaz iz programa:
main::x = 3
for::x = 5
for::x = 5
for::x = 5
main::x = 3
increase::a = 1
decrease::a = -1
main::a = 1
nonstatic::s=1
nonstatic::s=1
nonstatic::s=1
```

```
static::s=1
static::s=2
static::s=3
```

Primer 75 *Primer ilustruje vidljivost imena promenljivih.*

```
#include <stdio.h> main() {
    int pom=1;
    printf("Pre ulaska u unutrasnji blok pom=%d\n",pom);
    {
        int pom=50;
        printf("Pre izlaska iz unutrasnjeg bloka  pom=%d\n",pom);
    }
    printf("Nakon izlaska iz  unutrasnjeg bloka  pom=%d\n",pom);
}

Izlaz: Pre ulaska u unutrasnji blok pom=1
Pre izlaska iz unutrasnjeg bloka  pom=50
Nakon izlaska iz  unutrasnjeg bloka pom=1
```

7.3 Konverzija

7.3.1 Automatska konverzija

Ako je jedan od operanada različičit 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*/
```

7.3.2 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 76 *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*/
```

7.3.3 Funkcije koje vrše konverziju

Primer 77

```
#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 78 *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));
```

```
}

Izlaz:
Unesi neko veliko slovo:
J
Odgovarajuce malo slovo je j
```

Primer 79 *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);
}

Izlaz:

N je : 234
```

Primer 80 *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 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));

}

Izlaz:
Dekadna vrednost binarnog broja 11110000 je 240
Dekadna vrednost heksadekadnog broja FF je 255
```

Primer 81 *Program vrsi konverziju iz dekadnog brojnog sistema u datu osnovu*

```
#include<stdio.h>
#define OSNOVA 16
main()
{
    int x;  /* Broj cija se konverzija vrsi */
    int ostaci[32]; /* Niz ostataka pri deljenju sa osnovom */
    int i = 0;

        /* Unosi se dekadni broj */
    scanf("%d",&x);
```

```
    /* Srz algoritma konverzije */
    while(x>0)
    {
        /* novi ostatak se dodaje u pomocni niz */
        ostaci[i++] = x%OSNOVA;
        x/=OSNOVA;
    }

    /* Niz se ispisuje unatrag */
        for (i--; i>=0; i--)
        if (ostaci[i]<=10)  /* Slucaj kada je cifra dekadna */
            printf("%d",ostaci[i]);
        else    /* Slucaj kada cifra prevazilazi dekadni opseg */
            printf("%c",'A'+ostaci[i]-10);

    printf("\n");
}
```

7.4 #define sa argumentima

Primer 82 *Demonstracija pretprocesorske direktive* `#define` *sa argumentima*

```
#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 83 *Demonstracija pretprocesorske direktive* `#define` *sa argumentima*

```
#include <stdio.h>
#define KUBW(a) (a * a * a)
#define KUB(a)  ( (a) * (a) * (a))

main()

{

int b=1;

printf("KUB(%d) = %d\n", 2*b+4, KUBW(2*b+4) );
printf("KUB(%d) = %d\n", 2*b+4, KUB(2*b+4) );

}

Izlaz:
KUB(6) = 22
KUB(6) = 216
```

Primer 84 *Ilustacija beskonačne petlje:*

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

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

Primer 85

```
#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 i j za dva.

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

Takode treba voditi računa o zagradama. Sledeći makro prouzrokovaće neočekivane rezultate za poziv square(a+1)

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

Primer 86

```
#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
```

8

Programski jezik C

[1]

8.1 Sortiranje niza

Niz može biti sortiran ili uređjen u opadajućem, rastućem, neopadajućem i nerastućem poretku. Dat je algoritam za sortiranje niza koji se unosi sa ulaza u nerastućem poretku odnosno tako da važi da je `niz[0] >= niz[1] >= ... niz[n]`. *Jednostavnom modifikacijom ovog algoritma niz se može sortirati i u opadajućem, rastućem ili neopadajućem poretku.*

Primer 87 *Selection sort*
U prvom prolazu se razmenjuju vrednosti a[0] sa onim članovima ostatka niza koji su veći od njega. Na taj način će se posle prvog prolaza kroz niz a[0] postaviti na najveći element niza.

```
#include<stdio.h>
#define MAXDUZ 100

int main()
{
   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   /* Dimenzija niza, pomocna i brojacke promenljive */
   int n,pom,i,j;

   printf("Unsite dimenziju niza\n");
   scanf("%d",&n);

   if (n>MAXDUZ)
   {
        printf("Nedozvoljena vrednost za n\n");
        exit(1);
   }

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
```

[1] Zasnovano na primerima sa sajtova `http://www.matf.bg.ac.yu/~filip`,`http://www.matf.bg.ac.yu/~milena`.

```
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje*/
   for(i=0; i<n-1; i++)
      for(j=i+1; j<n; j++)
         if(a[i]<a[j])
         {
            pom=a[i];
            a[i]=a[j];
            a[j]=pom;
         }

   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   putchar('\n');

   return 0;

}
```

8.2 Enumeracija

```
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!!!*/
```

8.3 Strukture

Primer 88 *Napisati program koji izračunava obim i površinu trougla i kvadrata.*

```
/* Program uvodi strukture - geometrijske figure */
#include <stdio.h>

/* Zbog funkcije sqrt. */
#include <math.h>
/* Upozorenje : pod linux-om je potrebno program prevoditi sa
      gcc -lm primer.c
   kada god se koristi <math.h>
*/

/* Tacke su predstavljene sa dve koordinate. Strukturom gradimo novi tip podataka. */
struct point
{
    int x;
    int y;
};

/* Izracunava duzinu duzi zadatu sa dve tacke */
float segment_length(struct point A, struct point B)
{
    int dx = A.x - B.x;
    int dy = A.y - B.y;
    return sqrt(dx*dx + dy*dy);
}

/* Izracunava povrsinu trougla Heronovim obrascem.
   Argumenti funkcije su tri tacke koje predstavljaju temena trougla */
float Heron(struct point A, struct point B, struct point C)
{
    /* Duzine stranica */
    float a = segment_length(B, C);
    float b = segment_length(A, C);
    float c = segment_length(A, B);

    /* Poluobim */
    float s = (a+b+c)/2;

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

/* Izracunava obim poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float circumference(struct point polygon[], int num)
{
    int i;
    float o = 0.0;

    /* Dodajemo duzine stranica koje spajaju susedna temena */
    for (i = 0; i<num-1; i++)
        o += segment_length(polygon[i], polygon[i+1]);

    /* Dodajemo duzinu stranice koja spaja prvo i poslednje teme */
```

```
    o += segment_length(polygon[num-1], polygon[0]);

    return o;
}

/* Izracunava povsinu konveksnog poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float area(struct point polygon[], int num)
{
    /* Povrsina */
    float a = 0.0;
    int i;

    /* Poligon delimo na trouglove i posebno izracunavamo povrsinu svakoga od njih */
    for (i = 1; i < num -1; i++)
        a += Heron(polygon[0], polygon[i], polygon[i+1]);

    return a;
}

main()
{
    /* Definisemo dve promenljive tipa tacke */
    struct point a;

    /* Inicijalizujemo tacku b na (1,2) */
    struct point b = {1, 2};

    /* triangle je niz od tri tacke - trougao (0,0), (0,1), (1,0) */
    struct point triangle[3];

    /* square je niz od cetiri tacke - jedinicni kvadrat.
       Obratiti paznju na nacin inicijalizacije niza struktura */
    struct point square[4] = {{0, 0}, {0, 1}, {1, 1}, {1, 0}};

    /* Postavljamo vrednosti koordinata tacke a*/
    a.x = 0; a.y = 0;

    /* Gradimo trougao (0,0), (0,1), (1,0) */
    triangle[0].x = 0;  triangle[0].y = 0;
    triangle[1].x = 0;  triangle[1].y = 1;
    triangle[2].x = 1;  triangle[2].y = 0;


    /* Ispisujemo velicinu strukture tacka */
    printf("sizeof(struct point) = %d\n", sizeof(struct point));

    /* Ispisujemo vrednosti koordinata tacaka */
    printf("x koordinata tacke a je %d\n", a.x);
    printf("y koordinata tacke a je %d\n", a.y);
    printf("x koordinata tacke b je %d\n", b.x);
    printf("y koordinata tacke b je %d\n", b.y);
```

```
    printf("Obim trougla je %f\n",
        circumference(triangle, 3));
    printf("Obim kvadrata je %f\n",
        circumference(square, 4));
    printf("Povrsina trougla je %f\n",
        Heron(triangle[0], triangle[1], triangle[2]));
    /* Broj tacaka je moguce odrediti i putem sizeof */
    printf("Povrsina kvadrata je %f\n",
        area(square, sizeof(square)/sizeof(struct point)));

}

Izlaz:
sizeof(struct point) = 8
x koordinata tacke a je 0
y koordinata tacke a je 0
x koordinata tacke b je 1
y koordinata tacke b je 2
Obim trougla je 3.414214
Obim kvadrata je 4.000000
Povrsina trougla je 0.500000
Povrsina kvadrata je 1.000000
```

Primer 89 *Ilustracija korisšćenja* `typedef`.

```
/* Koriscenje typedef radi lakseg rada */
#include <stdio.h>

#include <math.h>
/* Ovim se omogucava da se nadalje u programu umesto int moze
   koristiti ceo_broj */
typedef int ceo_broj ;

/* Ovim se omogucuje da se nadalje u programu umesto struct point
  moze koristiti POINT */
typedef struct point POINT;

struct point
{
    int x;
    int y;
};

main()
{
    /* Umesto int mozemo koristiti ceo_broj */
    ceo_broj x = 3;

    /* Definisemo promenljivu tipa tacke.
       Umesto struct point mozemo koristiti POINT */
    POINT a;
```

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

    /* Postavljamo vrednosti koordinata tacke a*/
    a.x = 1; a.y = 2;
    /* Ispisujemo velicinu strukture tacka */
    printf("sizeof(struct point) = %d\n", sizeof(POINT));

    /* Ispisujemo vrednosti koordinata tacaka */
    printf("x koordinata tacke a je %d\n", a.x);
    printf("y koordinata tacke a je %d\n", a.y);

}

Izlaz:
x = 3
sizeof(struct point) = 8
x koordinata tacke a je 1
y koordinata tacke a je 2
```

Primer 90 *Strukture se u funkcije prenose po vrednosti. Moguće je koristiti pokazivače na strukture.*

```
#include <stdio.h>

typedef struct point
{
    int x, y;
} POINT;

/* Zbog prenosa po vrednosti tacka ne moze biti ucitana */
void get_point_wrong(POINT p)
{
    printf("x = ");
    scanf("%d", &p.x);
    printf("y = ");
    scanf("%d", &p.y);
}

/* Koriscenjem prenosa preko pokazivaca, uspevamo */
void get_point(POINT* p)
{
    /* p->x je skraceni zapis za (*p).x */

    printf("x = ");
    scanf("%d", &p->x);
    printf("y = ");
    scanf("%d", &p->y);
}


main()
{
    POINT a = {0, 0};
```

```
    printf("get_point_wrong\n");
    get_point_wrong(a);
    printf("a: x = %d, y = %d\n", a.x, a.y);

    printf("get_point\n");
    get_point(&a);
    printf("a: x = %d, y = %d\n", a.x, a.y);

}
```

8.4 Rad sa datotekama

1.
```
/* Program demonstrira otvaranje datoteka ("r" - read i "w" - write mod) i osnovne
   tehnike rada sa datotekama */

/* U datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke
   citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz */

#include <stdio.h>

/* Zbog funkcije exit */
#include <stdlib.h>

main()
{
        int i;
        int br;

        /* Otvaramo datoteku sa imenom podaci.txt za pisanje */
        FILE* f = fopen("podaci.txt", "w");

        /* Ukoliko otvaranje nije uspelo, fopen vraca NULL. U tom slucaju,
           prijavljujemo gresku i zavrsavamo program */
        if (f == NULL)
        {
                printf("Greska prilikom otvaranja datoteke podaci.txt za pisanje\n");
                exit(1);
        }

        /* Upisujemo u datoteku prvih 10 prirodnih brojeva (svaki u posebnom redu) */
        for (i = 0; i<10; i++)
                fprintf(f, "%d\n", i);

        /* Zatvaramo datoteku */
        fclose(f);

        /* Otvaramo datoteku sa imenom podaci.txt za citanje */
        f = fopen("podaci.txt", "r");

        /* Ukoliko otvaranje nije uspelo, fopen vraca NULL. U tom slucaju,
```

```
           prijavljujemo gresku i zavrsavamo program */
        if (f == NULL)
        {
                printf("Greska prilikom otvaranja datoteke podaci.txt za citanje\n");
                exit(1);
        }

        /* Citamo brojeve iz datoteke dok ne stignemo do kraja i ispisujemo ih
           na standardni izlaz */

                /* Pokusavamo da procitamo broj */
                while(fscanf(f, "%d", &br) == 1)
                    /* Ispisujemo procitani broj */
                    printf("Procitano : %d\n", br);

        /* Zatvaramo datoteku */
        fclose(f);

}
```

2.
```
/* Program demonstrira "a" - append mod datoteka - nadovezivanje */
#include <stdio.h>

main()
{
        FILE* datoteka;

        /* Otvaramo datoteku za nadovezivanje i proveravamo da li je doslo do greske */
        if ( (datoteka=fopen("dat.txt","a"))==NULL)
        {
                fprintf(stderr,"Greska : nisam uspeo da otvorim dat.txt\n");
                return 1;
        }

        /* Upisujemo sadrzaj  u datoteku */
        fprintf(datoteka,"Zdravo svima\n");

        /* Zatvaramo datoteku */
        fclose(datoteka);
}
```

3.
```
/* Program ilustruje rad sa datotekama. Program kopira
datoteku ulaz.txt u datoteku izlaz.txt. */
/* Uz svaku liniju se zapisuje i njen broj */
#include <stdio.h>

#define MAX_LINE 256

/* Funkcija getline iz K&R jednostavno realizovana preko funkcije fgets */

int getline(char s[], int lim)
{
        char* c = fgets(s, lim, stdin);
```

```
        return c==NULL ? 0 : strlen(s);
}

main()
{
        char line[MAX_LINE];
        FILE *in, *out;
        int line_num;

        if ((in = fopen("ulaz.txt","r")) == NULL)
        {
                fprintf(stderr, "Neuspesno otvaranje datoteke %s\n", "ulaz.txt");
                return 1;
        }

        if ((out = fopen("izlaz.txt","w")) == NULL)
        {
                fprintf(stderr, "Neuspesno otvaranje datoteke %s\n","izlaz.txt");
                return 1;
        }

        /*  Prepisivanje karakter po karakter je moguce ostvariti preko:
                int c;
                while ((c=fgetc(in)) != EOF)
                        putc(c,out);
        */

        line_num = 1;
        /* Citamo liniju po liniju sa ulaza*/
        while (fgets(line, MAX_LINE, in) != NULL)
        {
                /* Ispisujemo broj linije i sadrzaj linije na izlaz */
                fprintf(out, "%-3d :\t", line_num++);
                fputs(line, out);
        }

        /* Zatvaramo datoteke */
        fclose(in);
        fclose(out);
}
```

4.
```
/* Citanje niza struktura iz tektsualne datoteke - artikli prodavnice */

/* Datoteka cije se ime unosi sa standardnog ulaza sadrzi podatke o
   proizvodima koji se prodaju u okviru odredjene prodavnice.
   Svaki proizvod se odlikuje sledecim podacima :
        bar-kod   - petocifreni pozitivan broj
        ime       - niska karaktera
        cena      - realan broj zaokruzen na dve decimale
        pdv       - stopa poreza - realan broj zaokruzen na dve decimale
   Pretpostavljamo da su podaci u datoteci korektno zadati.
```

```
   Pretpostavljamo da se u prodavnici ne prodaje vise od 1000 razlicitih artikala.
   Na standardni izlaz ispisati podatke o svim proizvodima koji se prodaju.
*/

#include <stdio.h>

/* Maksimalna duzina imena proizvoda */
#define MAX_IME 30

/* Struktura za cuvanje podataka o jednom artiklu */
typedef struct _artikal
{
        int bar_kod;
        char ime[MAX_IME];
        float cena;
        float pdv;
} artikal;

/* Maksimalni broj artikala */
#define MAX_ARTIKALA 1000

/* Niz struktura u kome se cuvaju podaci o artiklima */
artikal artikli[MAX_ARTIKALA];

/* Broj trenutno ucitanih artikala */
int br_artikala = 0;

/* Ucitava podatke o jednom artiklu iz date datoteke.
   Vraca da li su podaci uspesno procitani */
int ucitaj_artikal(FILE* f, artikal* a)
{
        /* Citamo podatke */
         if((fscanf(f, "%d", &(a->bar_kod))==1)
         && (fscanf(f, "%s", a->ime)==1)
         && (fscanf(f, "%f", &(a->cena))==1)
         && (fscanf(f, "%f", &(a->pdv))==1))
            /* Prijavljujemo uspeh */
            return 1;
        else
            /* Prijavljujemo neuspeh. */
            return 0;
}

/* Izracunava ukupnu cenu datog artikla */
float cena(artikal a)
{
        return  a.cena*(1+a.pdv);
}

/* Ispisuje podatke o svim artiklima */
void ispisi_artikle()
{
```

```
        int i;
        for (i = 0; i<br_artikala; i++)
                printf("%-5d  %-10s %.2f %.2f    = %.2f\n",
                        artikli[i].bar_kod, artikli[i].ime,
                        artikli[i].cena, artikli[i].pdv, cena(artikli[i]));
}

main()
{
        FILE* f;


        /* Ucitavamo ime datoteke */
        char ime_datoteke[256];
        printf("U kojoj datoteci se nalaze podaci o proizvodima: ");
        scanf("%s", ime_datoteke);

        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (f = fopen(ime_datoteke, "r")) == NULL)
        {
        printf("Greska : datoteka %s ne moze biti otvorena\n",
                                        ime_datoteke);
        }


        /* Ucitavamo artikle */
        while (ucitaj_artikal(f, &artikli[br_artikala]))
                br_artikala++;

        /* Ispisujemo podatke o svim artiklima */
        ispisi_artikle();

        /* Zatvaramo datoteku */
        fclose(f);
}
```

Primer 91 *Program ilustruje čitanje etiketa iz neke HTML datoteke.*

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

/* Maksimalna duzina etikete */
#define MAX_TAG 100

#define OTVORENA 1
#define ZATVORENA 2
#define GRESKA 0

/* Funkcija ucitava sledecu etiketu
   i smesta njen naziv u niz s duzine max.
   Vraca OTVORENA za otvorenu etiketu,
   ZATVORENA za zatvorenu etiketu,
   odnosno GRESKA inace */
```

```
int gettag(FILE *f, char s[], int max)
{   int c, i;
    int zatvorenost=OTVORENA;

    /* Preskacemo sve do znaka '<' */
    while ((c=fgetc(f))!=EOF && c!='<')
                    ;
    /* Nismo naisli na etiketu */
    if (c==EOF)
           return GRESKA;

    /* Proveravamo da li je etiketa zatvorena */
    if ((c=fgetc(f))=='/')
       zatvorenost=ZATVORENA;
    else
       ungetc(c,f);

    /* Citamo etiketu dok nailaze slova
    i smestamo ih u nisku */
    for (i=0; isalpha(c=fgetc(f))
                    && i<max-1; s[i++] = c)
                     ;
    /* Vracamo poslednji karakter na ulaz
       jer je to bio neki karakter koji nije
       slovo*/
    ungetc(c,f);

    s[i]='\0';

    /* Preskacemo atribute do znaka > */
    while ((c=fgetc(f))!=EOF && c!='>')
    ;

    /* Greska ukoliko nismo naisli na '>' */
    return c=='>' ? zatvorenost : GRESKA;
}

main()
{
    char tag[MAX_TAG];
    int zatvorenost;

 FILE* f;


        /* Ucitavamo ime datoteke */
        char ime_datoteke[256];
        printf("Unesite naziv html dokumenta iz kog se vrsi citanje etiketa: ");
        scanf("%s", ime_datoteke);

        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (f = fopen(ime_datoteke, "r")) == NULL)
```

```
        {
        printf("Greska : datoteka %s ne moze biti otvorena\n",
                                        ime_datoteke);
        }

    while ((zatvorenost = gettag(f,tag,MAX_TAG))>0)
    {
    if (zatvorenost==OTVORENA)
        printf("Otvoreno : %s\n",tag);
        else
        printf("Zatvoreno : %s\n",tag);
    }

fclose(f);
}
```

Primer 92 *Sa standardnog ulaza se učitava niz od* `n (n<100)` *tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka odredjuje konveksni mnogougao i rezultat ispisati na standardni izlaz.*

```
#include<stdio.h>

typedef struct tacka
{
int x;
int y;
} TACKA;

/* F-ja ispituje da li se tacke T3 i T4 nalaze sa iste strane prave
 odredjene tackama T1 i T2.*/
int SaIsteStranePrave(TACKA T1,TACKA T2, TACKA T3, TACKA T4)
{
    int t3 = (T3.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T3.x - T1.x);
    int t4 = (T4.y - T1.y)*(T2.x - T1.x) - (T2.y - T1.y) * (T4.x - T1.x);
    return (t3 * t4 > 0);
}

main()
{
    TACKA mnogougao[100];
    int j,i;
    int n;
    int konveksan = 1;

    do
    {
        printf("Unesite broj temena mnogougla:\n");
        scanf("%d",&n);
        if(n<3)
            printf("Greska! Suvise malo tacaka! Pokusajte ponovo!\n");
    }
    while(n<3);
```

```
    printf("Unesite koordinate temena mnogougla takve da nikoja tri
                                        temena nisu kolinearna!\n");
    for(i=0;i<n;i++)
    scanf("%d %d", &mnogougao[i].x, &mnogougao[i].y);

    /* Da bi mnogougao bio konveksan potrebno (i dovoljno) je da kada se
    povuce prava kroz bilo koja dva susedna temena mnogougla sva ostala
    temena budu sa iste strane te prave.*/
    for(i=0;konveksan&&i<n-1;i++)
    {
        for(j=0;konveksan&&j<i-1;j++)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
        for(j=i+2;konveksan&&j<n-1;j++)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
        if(i!=0&&i!=n-1&&i+1!=0&&i+1!=n-1)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[0],mnogougao[n-1]);
    }
    for(j=1;konveksan&&j<n-2;j++)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[0],
                        mnogougao[n-1],mnogougao[j],mnogougao[j+1]);

    if(konveksan)
        printf("Uneti mnogougao jeste konveksan!\n");
    else
        printf("Uneti mnogougao nije konveksan!\n");

}
```

8.5 Formiranje HTML dokumenta

Primer 93 *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 94 *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 95 *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>");
return 0;
}
```

8.6 Argumenti komandne linije

Primer 96 *Ilustracija rada sa argumentima komandne linije.*

```
/* Program pozivati sa npr.:
        ./a.out
        ./a.out prvi
        ./a.out prvi drugi treci
```

```
        ./a.out -a -bc ime.txt
*/

#include <stdio.h>

/* Imena ovih promenljivih mogu biti proizvoljna. Npr.

        main (int br_argumenata, char* argumenti[]);

   ipak, uobicajeno je da se koriste sledeca imena:
*/

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

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

Primer 97 *Program ispisuje opcije navedene u komandnoj liniji.* K&R *rešenje.*

```
/* Opcije se navode koriscenjem znaka -,  pri cemu je moguce da iza jednog -
   sledi i nekoliko opcija.
   Npr. za -abc -d -fg su prisutne opcije a b c d f g  */

/* Resnje se intenzivno zasniva na pokazivackoj aritmetici i prioritetu operatora */

#include <stdio.h>

int main(int argc, char* argv[])
{
        char c;
        /* Dok jos ima argumenata i dok je karakter na poziciji 0 upravo crtica */
        while(--argc>0 && (*++argv)[0]=='-')
                /* Dok god ne dodjemo do kraja tekuceg stringa */
                while (c=*++argv[0])
                        printf("Prisutna opcija : %c\n",c);

}

Izlaz:
Prisutna opcija : a
Prisutna opcija : b
Prisutna opcija : c
Prisutna opcija : d
Prisutna opcija : f
Prisutna opcija : g
```

Primer 98 *Program ispisuje opcije navedene u komandnoj liniji - jednostavnija verzija.*

```
#include <stdio.h>

main(int argc, char* argv[])
{
        /* Za svaki argument komande linije, pocevsi od argv[1]
           (preskacemo ime programa) */
        int i;
        for (i = 1; i < argc; i++)
        {
                /* Ukoliko i-ti argument pocinje crticom */
                if (argv[i][0] == '-')
                {       /* Ispisujemo sva njegova slova pocevsi od pozicije 1 */
                        int j;
                        for (j = 1; argv[i][j] != '\0'; j++)
                                printf("Prisutna je opcija : %c\n", argv[i][j]);
                }
                /* Ukoliko ne pocinje crticom, prekidamo */
                else
                        break;
        }
}
```

Primer 99 *Iz datoteke čije se ime zadaje kao argrument komandne linije, učitati cele brojeve sve dok se ne učita nula, i njihov zbir ispisati u datoteku čije se ime takođe zadaje kao argument komandne linije.*

```
#include<stdio.h>
main(int argc, char* argv[])
{
    int n, S=0;
    FILE* ulaz, *izlaz;
    /* Ukoliko su imena datoteka navedena kao argumenti...*/
    if (argc>=3)
    {
        /* ...otvaramo datoteku i proveravamo da li smo uspeli */
         if ( (ulaz = fopen(argv[1], "r")) == NULL)
             printf("Greska : datoteka %s ne moze biti otvorena\n", argv[1]);
         if ( (izlaz = fopen(argv[2], "w")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", argv[2]);
    }
    else
    {
        char ime_datoteke_ulaz[256], ime_datoteke_izlaz[256];
        /* Ucitavamo ime datoteke */
        printf("U kojoj datoteci se nalaze brojevi: ");
        scanf("%s", ime_datoteke_ulaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (ulaz = fopen(ime_datoteke_ulaz, "r")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_ulaz);
        printf("U kojoj datoteci treba ispisati rezultat: ");
        scanf("%s", ime_datoteke_izlaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (izlaz = fopen(ime_datoteke_izlaz, "w")) == NULL)
```

```
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_izlaz);
    }

    fscanf(ulaz, "%d", &n);
    while(n!=0)
    {
        S+=n;
        fscanf(ulaz, "%d", &n);
    }
    fprintf(izlaz,"Suma brojeva ucitanih iz datoteke je %d.", S);
    return 0;
}
```

Primer 100 *Datoteka cije se ime unosi sa komandne linije sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

```
#include <stdio.h>

#define MAXL 100
#define MAXN 1000

typedef struct student {
    char ime[MAXL];
    char prezime[MAXL];
    short int indeks;
} student;

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

    FILE *in; int j,i=0;
    student niz[MAXN];

    if(argc!=2)
    {
        fprintf(stderr,"Neispravno pozivanje! Koriscenje: %s <ime datoteke>\n",argv[0]);
        return -1;
    }
    if(!(in=fopen(argv[1],"r")))
    {
        fprintf(stderr,"Ne mogu da otvorim datoteku %s za citanje.",argv[1]);
        return -1;
    }

    while(!feof(in))
    {
        fscanf(in,"%s %s %d",&niz[i].ime, &niz[i].prezime, &niz[i].indeks);
        i++;
    }

    /* Sredjujemo zadnji scanf koji ucitava EOF */
    i--;
```

```
    for(j=0;j<i;j++)
    {
        fprintf(stdout,"Ime: %s\nPrezime: %s\nIndeks: %d\n\n",
                        niz[j].ime, niz[j].prezime, niz[j].indeks);
    }

    fclose(in);
    return 0;
}
```

Zadaci za vežbu

Zadatak 38 *U datoteci brojevi.txt smeštena je prvo dimenzija niza a zatim i niz celih brojeva. Smatrati da nema više od 100 brojeva. Sa standardnog ulaza se učitava jedan ceo broj. Ispitati da li se taj broj nalazi u nizu brojeva učitanih iz datoteke brojevi.txt ili ne i rezultat ispisati na izlaz. Pri tome vršiti:*

a) linearnu

b) binarnu pretragu niza. Prethodno niz sortirati.

Zadatak 39 *Datoteka čije se ime unosi sa standardnog ulaza sadri podatke o uspehu studenata na kolokvijumima iz osnova programiranja. Prva linija datoteke sadrži broj studenata, a zatim svaka sledeća linija sadrži ime i prezime odredjenog studenta, njegov broj indeksa (u obliku korisničkog imena na alas-u npr. mr01123) i broj poena na prvom i na drugom kolokvijumu.*

a) Definisati strukturu podataka za čuvanje podataka o studentima

b) Učitati iz datoteke studente i smestiti ih u niz struktura. Ispisati taj niz studenata na standardni izlaz radi provere ispravnosti učitavanja niza.

c) Sortirati niz studenata u u opadajućem poretku prema broju poena.

d) U datoteku RezultatIspita.txt uneti spisak studenata koji su položili ispit sortiran u opadajućem poretku prema broju poena.

Ispit su položili samo oni kojima je zbir poena na prvom i drugom kolokvijumu bar pedeset.

Zadatak 40 *Napisati program koji iz datoteke čije se ime unosi sa standardnog ulaza, učitava niz struktura tačaka, izračunava obim poligona određen učitanim nizom tačaka i ispisuje njegovu vrednost na standardni izlaz. Smatrati da u datoteci nema više od 100 tačaka.*