Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

December 25, 2005

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/~jelenagr`, `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.
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
```

1.5.1 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
```

1.5.2 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.

1.5.3 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 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 ogičkih i relacijskih operatora.*

```
#include <stdio.h>

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

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

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

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
```

1.7.5 Switch

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

Primer 22 *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");
      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
```

1.8 Uslovni izraz

Slično kao `if`.

```
izraz1 ? izraz2 : izraz3

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

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 Učitavanje i ispis na izlaz - funkcije printf() i scanf()

Primer 23 *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 24 *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 25 *A iz ovog?*

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

Izlaz iz programa:
Slova
  z
    Z
```

Primer 26

```
#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 27 *Program prikazuje unos i ispis realnih brojeva.*

```
#include <stdio.h>
main()
{
    float x;
    printf("Unesi realan broj : ");

    /* Eksperimentisati sa vrednoscu npr. 34.56 */
    scanf("%f",&x);
    printf("Uneli ste broj (%%f): %f\n", x);
```

```
    printf("U naucnoj notaciji (%%g): %g\n", x);
}
Ulaz:
Unesi realan broj: 1.234
Izlaz iz programa:
Uneli ste broj (%f): 1.234000
U naucnoj notaciji (%g): 1.234
```

Primer 28 *Program prikazuje zaukruživanje realnog broja prilikom ispisa.*

```
#include <stdio.h>

main()
{
    float x;
    printf("Unesi realan broj : ");
    scanf("%f",&x);
    printf("Broj zaokruzen na dve decimale je : %.2f\n", x);
}

Ulaz:
Unesi realan broj: 1.234
Izlaz iz programa:
Broj zaokruzen na dve decimale je : 1.23
```

Primer 29 *Program ispisuje ascii tabelu.*

```
#include <stdio.h>

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

Primer 30 *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 31 *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 32 *Program menja mesta cifara u broju.*

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

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

2.2 Oblast važenja lokalnih promenljivih

Primer 33

```
#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
```

2.3 Operator sizeof

Primer 34 *Demonstracija sizeof operatora. sizeof operator izračunava veličinu tipa odnosno promenjive.*

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

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

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

}

Izlaz iz programa(u konkretnom slucaju):
sizeof(int)=4
sizeof(long)=4
sizeof(short)=2
sizeof(signed)=4
sizeof(unsigned)=4
sizeof(char)=1
sizeof(float)=4
sizeof(double)=8
sizeof(i)=4
sizeof(f)=4
sizeof(n)=40
Broj elemenata niza n : 10
```

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 3 cela broja sa ulaza.*

3

Programski jezik C

[1]

3.1 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!!!*/
```

3.2 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`.

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

Primer 35 *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 36 *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 37 *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 38 *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 39 *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 40 *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 41 *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 42 *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);
}
```

3.3 Nizovi

Deklaracija niza:

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

Pristupanje elementima niza:

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

Unos vrednosti elemenata niza sa tastature:

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

Stampanje elemenata niza

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

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

Primer 43 *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++)
    {   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 44 *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 45 *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]);

}
```

Zadaci za vežbu:

Zadatak 11 *U C-u nije precizirano da li je tip* `char` *označen ili ne.Š ta mislite o petlji oblika* `for(c=0;c<128;++c){...}` *na mašini na kojoj je* `char` *označen (*`signed`*)? Obrazložite!*

Zadatak 12 *Sastaviti logički izraz koji koji ispituje da li su dva cela broja x i y različite parnosti.*

Zadatak 13 *Diskutovati u zavisnosti od tipa promenljive c vrednost aritmetičkog izraza:*

```
int a = 7;
float b = 3.0;
c = (float) a / b;
```

Zadatak 14 *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 15 *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]);
```

Zadaci za praktikum:

Zadatak 16 *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);
}
```

Zadatak 17 *Prepisuje ulaz na izlaz čineći tabulatore, nove linije i backslash-ove vidljivim.*

```
#include <stdio.h>
main()
{
    int znak;
    znak=getchar();
    while( znak!=EOF )
      {
       if( znak=='\t' ) /*uciniti tab vidljivim */
       { putchar('\\'); putchar('t'); }
       else if( znak=='\n' ) /*uciniti new line vidljiv */
       { putchar('\\'); putchar('n'); putchar('\n'); }
       else if( znak=='\\' ) /*backslash udvojiti */
       { putchar('\\'); putchar('\\'); }
       else putchar(znak);

       znak=getchar();
       } /* while( znak!=EOF ) */

} /*main() */
```

Zadatak 18 *Program prepisuje standardni ulaz na standardni izlaz.*
Ilustracija redirekcije standardnog ulaza i izlaza :
pokrenuti program sa :

```
./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 19 *Program pronalazi maksimum brojeva sa ulaza - verzija sa nizom.*

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

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

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

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

Zadatak 20 *Program pronalazi maksimum brojeva sa ulaza - verzija bez niza.*

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

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

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

Zadatak 21 *Ispisati prvih 15 članova Fibonačijevog niza.*

```
#include <stdio.h>
    #define BROJ 15
```

```
    main()
    {
      int i; /*brojac u petlji */
      int fibonaci[BROJ]; /*niz koji cuva vrednosti iz f-lacije */

       /*inicijalizacije */
        fibonaci[0]=0;
        fibonaci[1]=1;

      /*formiranje vrednosti clana niza u zavisnosti od vrednosti prethodnika */
      for  (i=2;i<BROJ;++i)  fibonaci[i]=fibonaci[i-2]+fibonaci[i-1];

      /*ispis vrednosti clanova niza */
         for  (i=0;i<BROJ;++i)
                printf("%d ", fibonaci[i]);
}

Izlaz:
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

4

[1]

4.1 Ugnježdena petlja

Primer 46 *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 47 *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++) {
```

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

```
    /* 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 48 *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++;
 }
}
```

4.2 Bit-operatori

!!!Ne mešati sa logičkim operatorima!!!

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

Primer 49 *Demonstracija bitskih operatora*

```
#include <stdio.h>

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

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

Primer 50 *print_bits - stampa bitove u zapisu datog celog broja x.*

```
#include <stdio.h>

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

void print_bits(int x)
{
```

```
    /* Broj bitova tipa unsigned */
    int wl = sizeof(int)*8;

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

    putchar('\n');
}


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

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

Primer 51 *Program proverava da li se na k-tom mestu nalazi 1.*

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

Primer 52 *Program postavlja na k-to mesto 1*

```
#include <stdio.h>

void print_bits(int x);

main(){
  int n,k;
  printf("Unesite broj i poziciju tog broja koju zelite da proverite:\n");
  scanf("%d %d",&n,&k);
  printf("Binarno, une\v seni broj je\n");
  print_bits(n);
```

```
  printf("Novi broj je %d\n",(n |(1<<k)));
  printf("Binarno, novi broj je\n");
  print_bits((n |(1<<k)));
  return 0;
}
```

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

Primer 53 *Funkcija koja broji bitove postavljene na 1 u broju*

```
int bitcount(unsigned x)
{
    int b;
    for(b=0; x!=0; x>>=1)
    if (x & 01) b++;
    return b;
}
```

Primer 54 *sum_of_bits - izračunava sumu bitova datog neoznačenog broja.*

```
#include <stdio.h>

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

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

    putchar('\n');
}

/*
int sum_of_bits(unsigned x)
{

    int wl = sizeof(unsigned)*8;
    int br = 0;

    unsigned mask;
    for (mask = 1<<wl-1; mask; mask>>=1)
        if (x&mask)
            br++;

    return br;
}
```

```
*/

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

    return br;
}


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

Primer 55 *get_bits, set_bits, invert_bits - izdvajanje, postavljanje i invertovanje pojedinacnih bitova*

```
#include <stdio.h>

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

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

    putchar('\n');
}


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

```
            1111111...1100000
       a zatim komplementiramo
    */
    unsigned last_n_1 = ~(~0 << n);

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

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

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

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

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

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

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

}

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

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

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

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

```
Izlaz iz programa:
00001010101000001010111110100000
00000000000000000000000010101111
00001010101000001111111110100000
00001010101000000101000010100000
```

Primer 56 *right_rotate_bits, mirror_bits - rotiranje i simetrija bitova.*

```
#include <stdio.h>

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

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

    putchar('\n');
}


/* Funkcija vrsi rotaciju neoznacenog broja x za n pozicija u desno */
unsigned right_rotate(unsigned x, int n)
{
    int i;
    int wl = sizeof(unsigned)*8;

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

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

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

        x |= last_bit<<wl-1;
    }

    return x;
}

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

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

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

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

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

Izlaz iz programa:
11111010111100001111101011110000
00001111010111110000111101011111
00111110101111000011111010111100
```

Zadaci za vežbu:

Zadatak 22 *Napisati program koji ispituje da li dva niza imaju barem jedan zajednički element.*

Zadatak 23 *Napisati operator dodeljivanja koji će broju x tipa unsigned sačuvati n krajnjih desnih bitova, a ostale postaviti na nulu.*

```
x=x&~(~0 << n); ili x&=~(~0 << n);
```

Zadatak 24 *Napisati operator dodeljivanja koji će u x očistiti n bitova (postaviti nule) počev od pozicije p.*

```
x&=~(~(~0<<n)<<(p-1))
```

Zadatak 25 *Napisati operator dodeljivanja kojim se invertuje x (prevodi jedan u nulu i nula u jedan) počev od pozicije p na dužini n.*

```
x^=(~(~0<<n)<<(p-1));
```

Zadatak 26 *Program koji sabira pozitivne brojeve niza cifara koji se završava nulom.*

```
#include<stdio.h>

main()
```

```
{
 int x, zbir;

 printf("Unesite niz cifara pri cemu je 0 oznaka za kraj\n");

 /* Beskonacna while petlja. */
 while( 1 ) {
    /* Citamo sledeci element... */
    scanf("%d", &x );
    /* ako smo procitali 0 znaci da smo stigli do kraja... */
    if( x == 0 )
        /* i izlazimo iz petlje */
        break;
    /* Ako je broj negativan preskacemo ga... */
    else if( x < 0 )
        /* i idemo na sledeci */
        continue;
    /* inace, broj je pozitivan i dodajemo ga u zbir. */
    else zbir = zbir + x;
 }

 printf("Suma pozitivnih je %d\n", zbir);
}
```

Primer 57 *Program koji računa zbir* $1 + x + \frac{x^2}{2} + \ldots + \frac{x^n}{n!}$

```
#include<stdio.h>

main()
{
 float f, suma; /* Faktor sume i suma */
 float x;       /* Promenljiva x iz izraza */
 int i;         /* Brojac u petljama */
 int n;         /* Broj sabiraka */

 scanf("%d", n);
 scanf("%d", x);

 /* Pocetne inicijalizacije */
 f = 1;
 suma = 1;

 /* U jednom prolazu petlje dodajemo tekuci sabirak */
 for(i = 1; i <=n; i++) {
    f = f * x / i;
    suma = suma + f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

Primer 58 *Napisati program koji računa sumu* $x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n * \frac{x^{2n-1}}{(2n-1)!}$

```
#include<stdio.h>

main()
{
 float f, suma, x;
 int i, n;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 suma = x;
 f = x;

 for(i = 1; i <= n; i++) {
    f = -f * x * x / ((2*i+1)*2*i);
    suma = suma * f;
 }

 printf("Suma prvih %d clanova je %f \n", n, suma);
}
```

Primer 59 (DOMAĆI) *Napisati program koji računa sumu* $1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^n \frac{x^{2n}}{(2n)!}$

Primer 60 *Program koji računa sumu* $x - \frac{x^3}{3*1!} + \frac{x^5}{5*2!} - \frac{x^7}{7*3!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)*n!}$

```
#include<stdio.h>

main()
{
 int i, n;
 float x, f, suma;

 scanf("%d", &n);
 scanf("%f", &x);

 /* Pocetne inicijalizacije */
 f = x;
 suma = x;

 for(i = 1; i < n; i++) {
    f = -f * x * x / i;
    suma = suma + f/(2*i+1);
 }

 printf("Suma prvih %d clanoca je %f\n", n, suma);
}
```

5

[1]

5.1 Funkcije

Primer 61 *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 62 *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 63 *power - funkcija koja stepenuje realan broj na celobrojni izlozilac*

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

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

```
/* 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 64 *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 65 *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 66

```
#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
```

5.2 Lenjo izračunavanje

Primer 67 *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
```

Primer 68 *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 27 *Napisati f-ju koja za uneti broj n izračunava zbir recipročnih vrednosti prvih n brojeva.*

Zadatak 28 *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 29 *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.

6

[1]

6.1 Obrada teksta sa ulaza - putchar(), getchar()

Primer 69 *Brojanje reči, linija i karaktera sa ulaza*

```
#include <stdio.h>

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

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

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

state = OUT;

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

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

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

Primer 70 *Program prepisuje ulaz na izlaz pri čemu više blanko znakova zamenjuje jednim.*

```
#include <stdio.h>

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

6.2 Formiranje HTML dokumenta

Primer 71 *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 72 *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 73 *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;
}
```

6.3 Životni vek i oblast važenja promenjivih

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 *Ilustruje vidljivost imena*

```
#include <stdio.h>

int i=10;

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

6.4 Konstante

Koji su tipovi konstanti?

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

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

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

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

Posebni znaci su znak za kraj reda, tab i slično.

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

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

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

!!!Važno:

Koja je razlika između "x" i 'x'?

Primer 76 *Program uvodi niske karaktera terminisane nulom.*

```
#include <stdio.h>

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

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

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

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

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

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

Izlaz:
abc
abc
abc
```

```
abc
```

Primer 77 *Primer funkcije koja izračunava dužinu niske znakova.*

```
#include <stdio.h>

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

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

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

Izlaz:
Duzina ove niske je: 20
```

6.5 Konverzija

6.5.1 Automatska konverzija

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

Naredba dodele:

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

obrnuto:

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

6.5.2 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 78 *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*/
```

6.5.3 Funkcije koje vrše konverziju

Primer 79

```
#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 80 *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 81 *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 82 *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 83 *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");
}
```

Zadaci za vežbu

Zadatak 30 *1. Napisati funkciju koja računa k-ti stepen prirodnog broja n.*

2. Napisati program koji formira HTML dokument koji sadrži tabelu u čijoj se j-toj ćeliji i-te kolone nalazi vrednost i^j.

3. Napisati program koji za uneti niz koeficijenata $a[i]$ i uneti broj x računa vrednost polinoma $a_n * x^n + a_{n-1} * x^{n-1} + ... + a_1 * x + a_0$

Zadatak 31 *Napisati program koji formira HTML dokument koji sadrži numerisanu lista čija je svaka stavka malo slovo engleske abecede i njemu odgovarajući kod.*

Zadatak 32 *Napisati program koji broji pojavljivanja samoglasnika i suglasnika u tekstu koji se unosi sa standardnog ulaza (koristiti naredbu* `switch`*).*

Zadatak 33 *Napisati program koji broji pojavljivanja za svako od slova engleske abecede u tekstu koji se unosi sa standardnog ulaza i štampa rezultat na standardni izlaz.*

Zadatak 34 *1. Napisati funkciju koja konvertuje malo u veliko slovo.*

2. Napisati program koji prepisuje ulaz na izlaz pri čemu se sva mala slova konvertuju u velika.

7

[1]

7.1 Prenos parametara po vrednosti

Primer 84 *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
```

7.2 Prenos niza u f-ju

Primer 85 *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';
```

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

```
}

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

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

Primer 86 *Funkcija za ispis niza brojeva - demonstrira prenos nizova brojeva u funkciju.*

```
#include <stdio.h>

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

   Nizove je neophodno prenositi zajedno sa dimenzijom niza
   (osim 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 87 *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
```

Primer 88 *Program vrši linearnu pretragu niza.*

```
#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;
    return -1;
}

main()
{
    int a[] = {4, 3, 2, 6, 7, 9, 11};
    int br_elem = sizeof(a)/sizeof(int);
    int x;
    int i;
    printf("Unesi 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);
}
```

7.2.1 Funkcije za rad sa stringovima

Primer 89 *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 90 *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 91 *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 92

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

Primer 93 *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 94 *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 95 *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 35 *Napisati program koji u ucitanoj niski karaktera sa ulaza prebrojava pojavu cifara.*

Zadatak 36 *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 37 *Broj je* **Nivenov** *ako je deljiv sumom svojih cifara.*

1. *Napsati funkciju int sumaCifara(char a[]) koja računa sumu cifara broja* ***a****. Na primer, za "121" funkcija treba da vrati 4.*
2. *Napisati funkciju int Nivenov(char a[]) koja proverava da li je broj* ***a*** *Nivenov i vraća 1 ako jeste a 0 ako nije.*
3. *Napisati program koji sa ulaza učitava broj (kao nisku karaktera) i ispisuje na izlaz informaciju o tome da li je broj Nivenov.*

Zadatak 38 *Napisati program koji proverava da li je uneta niska palindrom. Na primer, palindrom je "anavolimilovana".*

8

[1]

8.1 Funkcije za rad sa stringovima

Primer 96 *Program pronalazi najdužu liniju sa ulaza.*

```
#include <stdio.h>
#define MAX_DUZINA_LINIJE 80

int ucitaj_liniju(char linija[])
{
    int c;
    int pozicija = 0;
    while ((c=getchar())!=EOF && c != '\n' &&
        pozicija<MAX_DUZINA_LINIJE-1)
        linija[pozicija++]=c;

    if (c=='\n')
        linija[pozicija++] = c;

    linija[pozicija]=0;

    return pozicija;
}

/*
int duzina_linije(char linija[])
{
    int d;
    for (d=0; linija[d]!=0; d++)
        ;

        d=0;
        while (linija[d] != 0)
            d++;

    return d;
```

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

```
}
*/

void kopiraj_liniju(char u[], char iz[])
{
    int pozicija;
    pozicija = 0;
    do
    {
        u[pozicija] = iz[pozicija];
        pozicija++;
    } while (u[pozicija-1]!=0);

    /*
    do
    {
        u[pozicija] = iz[pozicija];
    } while (u[pozicija++]!=0);
    */

    /*
        pozicija = 0;
        while((u[pozicija]=iz[pozicija])!=0)
            pozicija++;
    */

    /* while (a != 0) <==> while (a) */

    /* for (pozicija = 0; (u[pozicija]=iz[pozicija])!=0; pozicija++)
           ;
    */

    /* for (pozicija = 0; u[pozicija]=iz[pozicija]; pozicija++)
               ;
    */

}

main()
{
    char linija[MAX_DUZINA_LINIJE];
    char najduza_linija[MAX_DUZINA_LINIJE];

    int duzina_linije = 0;
    int duzina_najduze_linije = 0;

    najduza_linija[0] = 0;



    while( (duzina_linije = ucitaj_liniju(linija)) != 0)
    {
```

```
        if (duzina_linije>duzina_najduze_linije)
        {
            kopiraj_liniju(najduza_linija, linija);
            duzina_najduze_linije = duzina_linije;
        }
    }

    printf("Najduza linija je : %s\n",najduza_linija);
}
```

Primer 97 *Program pronalazi najduzu liniju sa ulaza(II verzija).*

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

#define MAX_DUZINA_LINIJE 80

int ucitaj_liniju(char linija[])
{
    int c;
    int pozicija = 0;
    while ((c=getchar())!=EOF &&
        c != '\n' &&
        pozicija<MAX_DUZINA_LINIJE-1)
        linija[pozicija++]=c;

    if (c=='\n')
        linija[pozicija++] = c;

    linija[pozicija]=0;

    return pozicija;
}

main()
{
    char linija[MAX_DUZINA_LINIJE];
    char najduza_linija[MAX_DUZINA_LINIJE];

    int duzina_linije = 0;
    int duzina_najduze_linije = 0;

    najduza_linija[0] = 0;



    while( (duzina_linije = ucitaj_liniju(linija)) != 0)
    {
        if (duzina_linije>duzina_najduze_linije)
        {
            strcpy(najduza_linija, linija);
            duzina_najduze_linije = duzina_linije;
        }
```

```
    }

    printf("Najduza linija je : %s\n",najduza_linija);
}
```

8.1.1 Linearna i binarna pretraga niza

Primer 98 *Linearno pretraživanje*

```
int linsearch(int x, int v[], int n)
{
    int i;
     for(i=0; i<n; i++)
         if(v[i]==x)
             return i;
     return -1; /* Nema poklapanja*/
}
```

Primer 99 *Binarno pretraživanje*

```
/* Napomena: Niz mora biti uredjen u rastucem redosledu. */
/* binsearch:  find x in v[0] <= v[1]
<= ... <= v[n-1] */
int binsearch(int x, int v[], int n)
{
    int low, high, mid;

    low = 0;
    high = n - 1;
    while (low <= high) {
        mid = (low+high)/2;
        if (x < v[mid])
            high = mid + 1;
        else if (x  > v[mid])
            low = mid + 1;
        else    /* found match */
            return mid;
    }
    return -1;   /* no match */
}
```

8.2 Makroi

Primer 100 *Šta je rezultat rada sledeceg programa :*

```
#include <stdio.h>
#define IN 1
#define OUT 0
main( )
{ int c, nw , state;
state=OUT;
```

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

```
Rezultat:
U sledecem novom redu se ispisuje broj reci teksta sa ulaza, pa se predje u novi red.
Smatramo da rec je ma koji niz znakova koji ne sadri blanko, tab, new line.
```

Primer 101 *Demonstracija pretprocesorske direktive* `#define`

```
#include<stdio.h>
/* Racuna sumu dva broja */
#define sum(a,b) ((a)+(b))

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

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

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

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

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

```
min(2,1,3) = 1
min(2,3,1) = 1
min(3,1,2) = 1
min(3,2,1) = 1
```

Primer 102 *Demonstracija pretprocesorske direktive* `#define`

```
#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 103 *Ilustacija beskonačne petlje:*

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

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

Primer 104

```
#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++)
```

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

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

Primer 105

```
#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
```

8.3 Pokazivači - osnovni pojmovi

Primer 106 *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);


    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
```

Primer 107 *swap : Demonstracija prenosa argumenata preko pokazivača.*

```
#include <stdio.h>

/* Pogresna verzija funkcije swap. Zbog prenosa po vrednosti, funkcija
   razmenjuje kopije promenljivih iz main-a, a ne samih promenljivih */
void swap_wrong(int x, int y)
{
    int tmp;

    printf("swap_wrong: ");
    printf("Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("x : %p\n", &x);
```

```
    printf("y : %p\n", &y);


    tmp = x;
    x = y;
    y = tmp;
}

/* Resenje je prenos argumenata preko pokazivaca */
void swap(int* px, int* py)
{
    int tmp;

    printf("swap : Funkcija menja vrednosti promenljivim na adresama : \n");
    printf("px = %p\n", px);
    printf("py = %p\n", py);

    tmp = *px;
    *px = *py;
    *py = tmp;
}

main()
{
    int x = 3, y = 5;
    printf("Adresa promenljive x je %p\n", &x);
    printf("Vrednost promenljive x je %d\n", x);
    printf("Adresa promenljive y je %p\n", &y);
    printf("Vrednost promenljive y je %d\n", y);

    /* Pokusavamo zamenu koristeci pogresnu verziju funkcije */
    swap_wrong(x, y);

    printf("Posle swap_wrong:\n");
    printf("Vrednost promenljive x je %d\n", x);
    printf("Vrednost promenljive y je %d\n", y);

    /* Vrsimo ispravnu zamenu. Funkciji swap saljemo adrese promenljvih
       x i y, a ne njihove vrednosti */
    swap(&x, &y);

    printf("Posle swap:\n");
    printf("Vrednost promenljive x je %d\n", x);
    printf("Vrednost promenljive y je %d\n", y);
}

Izlaz u konkretnom slucaju:
Adresa promenljive x je 0012FF88
Vrednost promenljive x je 3
Adresa promenljive y je 0012FF84
Vrednost promenljive y je 5
swap_wrong: Funkcija menja vrednosti promenljivim na adresama :
```

```
x : 0012FF78
y : 0012FF7C
Posle swap_wrong:
Vrednost promenljive x je 3
Vrednost promenljive y je 5
swap : Funkcija menja vrednosti promenljivim na adresama :
px = 0012FF88
py = 0012FF84
Posle swap:
Vrednost promenljive x je 5
Vrednost promenljive y je 3
```

Primer 108 *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
```

Zadaci sa vežbi:

Zadatak 39 *Sastaviti C-direktivu koja odre duje srednji po vrednosti od tri broja.*

Zadatak 40 *Sastaviti C-direktivu koja implementira operator implikacije.*

Zadatak 41 *Učitava se linija po linija teksta. Odštampati svaku od tih linija tako da ima veliko slovo na početku rečenice i sva mala unutar rečenice(., ?, !).*

Zadatak 42 *Napisati funkciju koja izračunava celobrojni koren prirodnog broja i program koji tu funkciju poziva za vrednost unetu sa standardnog ulaza.*

Zadatak 43 *Napisati funkciju koja izračunava celobrojni količnik dva uneta cela broja i program koji tu funkciju poziva za vrednosti unete sa standardnog ulaza.*

Zadatak 44 *Napisati funkciju koja izračunava NZD dva uneta broja koristeći Euklidov algoritam i program koji tu funkciju poziva za vrednosti unete sa standardnog ulaza.*

Zadatak 45 *Napisati funkciju koja izračunava celobrojni koren prirodnog broja i program koji tu funkciju poziva za vrednost unetu sa standardnog ulaza.*

9

9.1 Zadaci sa prethodnih ispita i kolokvijuma

Zadatak 46 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 47 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 48 I kolokvijum, februar 2005. *Napisati program koje će učitati sa tastature broj* `s(unsigned int)` *i brojeve* `m` *i* `n(int)`*, pri čemu je* `0<=m<=n<sizeof(unsigned)*8` *i formirati vrednost* `d` *(unsigned int) u kojoj je bit na poziciji* `i` *jednak 1 akko je* `m <=i<=n`*(pozicije se broje od nule sdesna na levo).Program treba da na standardnom izlazu ispiše broj koji se dobija od s postavljanjem na 0 svih bitova koji su u d jednaki 1.*

Zadatak 49 Septembar, 2005. *Napisati program koji sa standardnog ulaza učitava pozitivan ceo broj, a na standardni izlaz ispisuje vrednost tog broja sa razmenjenim vrednostima bitova na poziciji* `i, j`*. Pozicije i, j se učitavaju kao parametri komandne linije. Smatrati da krajnji desni bit binarne reprezentacije je 0-ti bit. Pri rešavanju nije dozvoljeno koristiti pomoćni niz niti aritmetičke operatore* `+,-,/,*,%`*.*

Zadatak 50 Septembar, 2005. *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.*

Zadatak 51 Jun, 2004. *Datoteka Matrice.txt sadrži dve celobrojne kvadratne matrice. U datoteci su prvo zapisane dimenzije matrica* `n i m (n > m)` *a zatim i elementi prvo jedne a zatim i druge matrice. Napisati program koji proverava da li se manja matrica sadrži u većoj. Matrica se sadrži u matrici veće dimenzije ukoliko postoji podmatrica veće matrice identična manjoj matrici tj. ako postoji blok veće matrice dimenzije* `m x m` *čiji su elementi jednaki elementima manje matrice na odgovarajućim pozicijama. npr. U matrici*

```
1 1 1
2 2 2
3 3 3
```

se sadrži matrica

```
1 1
2 2
```

a ne sadrži matrica

```
1 1
3 3
```

Zadatak 52 Jun, 2004. *Napisati funkciju koja računa multiplikativnu otpornost datog pozitivnog broja. Multiplikativna otpornost se računa na sledeći način* `n0 = n, nk` *je jednak proizvodu cifara broja* `n k-1, k = 1, 2 . . .` *, multiplikativna otpornost je najmanje* `k` *za koje je* `nk` *jednocifren broj. Napisati program koji iz datoteke čije se ime zadaje kao prvi argument komandne linije čita brojeve, gde su brojevi zapisani po jedan u svakom redu i u drugu datoteku čije se ime zadaje kao drugi argument komandne linije upisuje red po red date brojeve i njihovu multiplikativnu otpornost.*

Zadatak 53 Prvi kolokvijum za II tok 2004.godine - rad na racunaru *Napisati program koji generiše* `HTML` *fajl Boje.html koji sadrži tabelu boja. Tabela treba da ima 8 kolona pri čemu ćelije neparnih kolona treba da sadrže heksadekadnu vrednost boje i to u formatu* `ROGOBO` *a ćelije odgovarajuće parne kolone treba da budu obojene tom bojom.*

Zadatak 54 Prvi kolokvijum za II tok 2004.godine - rad na racunaru *Sa standardnog ulaza se unose veliki, celi, neoznačeni brojevi sa najviše 100 cifara. Ovih brojeva ima manje od 100 ali njihov broj nije unapred poznat. Napisati program koji sabira ovako unete brojeve i na standardni izlaz ispisuje njihov zbir.*
Napomena : Svaki broj se unosi u posebnom redu a potrebno je voditi računa o korektnosti ulaznih podataka.

Zadatak 55 Prvi kolokvijum za II tok 2004.godine - rad na papiru *Sa standardnog ulaza se unose dve niske koje predstavljaju elemente dva skupa. Skupovi nemaju više od 20 elemenata. Napisati program koji na standardni izlaz ispisuje niske koje predstavljaju:*

1. *presek,*
2. *uniju i*
3. *razliku*

elemenata dva skupa.

Zadatak 56 Januar, 2002. *Datoteka "izrazi.dat" sadrži izraze koji se sastoje od celobrojnih i realnih konstanti i operacija* `+,-,*, /` *i zapisani su u inverznoj poljskoj notaciji (operandi pa operacija). Na primer, izraz* `(1+2)/(3-4)` *zapisan je kao* `1 2 + 3 4 - /`*, a izraz* `21+7*6` *kao* `21 7 6 * +`*. Svaki izraz je u datoteci zapisan u novom redu i podrazumeva se da su izrazi sintaksno ispravni. Napisati program koji izračunava i štampa na ekran vrednosti svih izraza u datoteci. Rešenje napisati modularno i obavezno ga komentarisati.*

Zadatak 57 Januar, 2002. *Program sa standardnog ulaza učitava raspored 8 topova na šahovskoj tabli. Raspored se sastoji od 8 linija sa po 8 brojeva u svakoj liniji. Svaka linija odgovara jednom redu table, a svaki broj jednom polju. Broj ima vrednost 0 ako na datom polju nema topa i vrednost 1 ako na datom polju postoji top. Program treba da ispita da li je uneseni raspored validan (tj. da li je svaki učitani broj 1 ili 0 i da li ima ukupno 8 topova na tabli), kao i da odredi da li se u datom rasporedu neka dva topa tuku (topovi se tuku ukoliko se nalaze u istom redu ili istoj koloni table). Program treba da ispiše na standardnom izlazu "raspored nije validan" ukoliko ulazni podaci nisu dobri, a u suprotnom "ne tuku se" ukoliko je raspored takav da se nijedan par topova međusobno ne tuče, odn. "tuku se" ukoliko ima topova koji se tuku*

Zadatak 58 Januar, 2002. *Sa standardnog ulaza se učitava u jednoj liniji prirodan broj n, a potom i linije teksta do markera kraja fajla. Napisati program koji štampa n reči koje se najčešće pojavljuju i to počev od najfrekventije reči. Uz reč odštampati i broj pojava. Reč je po definiciji ma koji niz karaktera koji ne sadrži blanko, tabulator, znak za novi red. Sve poruke o greškama ispisati na standardnom izlazu za poruke o grešci.*

Zadatak 59 Januar, 2002. *Napisati program koji čita ulaznu datoteku ulaz.htm i štampa na standardni izlaz samo linije koje imaju 70 karaktera van etiketa, pri čemu se tekst markiran u obliku* `&entity;` *(npr.* `<` `&`*) ili* `&#number;` *(npr.* `&#269;`*) broji kao 1 karakter. Programi da budu pisani čitko i izdašno komentarisani.*

Zadatak 60 Februar, 2002. *Neka se relacija nad nekim skupom elemenata opisuje kvadratnom matricom na sledeći način: ako je u preseku* `i`*-te vrste i* `j`*-te kolone 1, to znači da je* `i`*-ti element u relaciji sa* `j`*-tim, ako je 0 to znači da nije u relaciji. Sa standardnog ulaza zadaje se najpre dimenzija ovakve matrice, pa zatim elementi matrice, jedan za drugim, po vrstama. Dimenzija matrice nije ograničena. napisati program koji, pošto proveri korektnost ulaza, za ovako zadatu relaciju ispituje njenu refleksivnost, simetričnost i tranzitivnost i odgovarajuće poruke štampa na ekran.*

Zadatak 61 Februar, 2002. *Datoteka prica.txt sadrži niz reči (reč je niz karaktera koji ne sadrži blanko, tabulator ili znak za novi red). Sa standardnog ulaza učitava se jedna reč. Nijedna reč, nema više od 20 karaktera. Napisati program koji broji i štampa na ekran koliko se puta data reč pojavila u datoteci, ako se zna da su neke reči pogrešno unete. Smatramo da je neka reč jednaka učitanoj i onda kada:*
- je zamenjeno jedno slovo nekim drugim slovom
- ili je izostavljeno jedno slovo u jednoj od te dve reči

Zadatak 62 Februar, 2002. *Napisati program koji za dva data pravougaonika* `R0` *i* `R1` *sa stranicama paralelnim koordinatnim osama izračunava i na standardni izlaz ispisuje površine njihovih unija* `(R0 R1)`*, presjeka* `(R0 R1)` *i razlike* `(R0 \ R1)`*. Pravougaonici se učitavaju sa standardnog ulaza i zadati su koordinatama donjeg lijevog, odn. gornjeg desnog tjemena. Ove koordinate su realni brojevi. Za čuvanje podataka koji određuju neki pravougaonik deklarisati odgovarajuću strukturu. Sve operacije nad pravougaonikom (ili pravougaonicima) izdvojiti u posebne funkcije. Primjer: za pravougaonike zadate na sledeći način:*

```
10 20 30 40
20 30 40 50
```

program treba da ispiše:

```
Povrsina unije iznosi 700
Povrsina preseka iznosi 100
Povrsina razlike iznosi 300"
```

Zadatak 63 Februar, 2002. *U datoteci tajna.txt nalazi se riječ dužine ne veće od 20 karaktera. Riječ se sastoji isključivo od malih slova. Napisati program za pogađanje riječi. Program treba da učita riječ iz datoteke, a zatim da sa standardnog ulaza čita jedno po jedno slovo koja daje korisnik pogađajući da li ih riječ sadrži. Po učitavanju svakog slova program treba da ispiše ona slova u riječi koja su dotad pogođena. Na mijestima ostalih slova treba da budu karakteri *. Voditi računa o mogućnosti da korisnik greškom unese nešto što nije slovo, takođe i neko slovo koje je ranije već unosio. Program ne treba da pravi razliku između malih i velikih slova, tj. ako korisnik unese neko veliko slovo, program treba da ga tretira kao malo slovo. Kada sva slova budu pogođena, program treba da ispiše ukupan broj pokušaja. Primjer sesije za slučaj kada je riječ koja se pogađa* ***zdravo*** *bi mogao biti:*

```
a
***a**
e
***a**
i
***a**
o
***a*o
r
**ra*o
m
**ra*o
b
**ra*o
d
*dra*o
v
*dravo
z
zdravo
Ukupan broj pokusaja: 10
```

Zadatak 64 Februar, 2002. *Napisati program koji učitava kvadratnu matricu sa standardnog ulaza čiji su članovi celi brojevi i proverava da li je matrica ortogonalna. Ne koristiti pomoćne matrice! U prvoj liniji nalaze se dimenzija matrice, a zatim se u svakoj liniji nalaze vrste matrice. Elementi unutar vrste su razdvojeni blanko znakovima. Dimenzija matrice nije unapred poznata. Pretpostaviti da su sve linije sem prve u ispravnom formatu i u slučaju greške izdati poruku na standardnom izlazu za poruke o grešci.*

Zadatak 65 Februar, 2002. *Parametri komandne linije su imena dve datoteke i ceo broj n. Napisati program koji poslednjih n linija prve datoteke upisuje u drugu datoteku. Može se pretpostaviti da prva datoteka ne sadrži linije duže od 80 karaktera, ali broj linija u datoteci nije unapred ograničen. U slučaju greške izdati poruku na standardnom izlazu za poruke o grešci.*

Programe komentarisati i programski kod pisati čitko.

Zadatak 66 April, 2002. *. Prvi red standardne ulazne datoteke sadrži 2 cela broja manja od 50 koji predstavljaju redom broj vrsta i broj kolona realne matrice* `A`*. Svaki sledeći red sadrži po jednu vrstu matrice. Napisati program koji :*

1. *nalazi sve elemente matrice A koji su jednaki zbiru svih svojih susednih elemenata i štampa ih u obliku (broj vrste, broj kolone, vrednost elementa)*
2. *nalazi i štampa sve četvorke oblika* `( A(i,j), A(i+1,j), A(i,j+1),A(i+1,j+1) )` *u kojima su svi elementi međusobno različiti.*

Zadatak 67 April, 2002. *Parametri komandne linije su nazivi 2 datoteke. Prva datoteka sadrži niz reči čiji broj i čija dužina nije ograničena (mogu biti proizvoljno veliki brojevi) . Reč je bilo kakav niz karaktera koji nije blanko, tabulator ili oznaka za kraj reda. Napisati program koji u drugu datoteku prepisuje samo one reči iz prve datoteke koje su parne dužine i koje i počinju i završavaju se slovom. (napomena: obavezno voditi računa o tome da se dužina reči ne moze ograničiti!)*

Zadatak 68 April, 2002. *Napisati program koji ispisuje kalendar za zadati mjesec i godinu* `XX` *vijeka. Poznato je da je* `1. januar 1901.` *bio utorak. Program prima dva argumenta u komandnoj*

liniji: broj u intervalu `[1, 12]` *koji predstavlja mjesec i broj u intervalu* `[1901, 2000]` *koji predstavlja godinu (obavezno proveriti validnost ovih argumenata). Program treba da ispiše kalendar na standardni izlaz i to tako što će u prvom redu biti ispisani mjesec (punim imenom) i godina, u narednom redu dvoslovne skraćenice od imena dana, počev od ponedeljka i sa po jednim blanko znakom između skraćenica, a zatim u narednim redovima datumi, pri čemu se za svaki dan odvajaju po 2 mjesta u kojima broj treba da je poravnat udesno, a između dana se ostavlja po jedan blanko znak. Tako npr, ako su argumenti koji su zadati u komandnoj liniji* `1 1970`*, ispis treba da ima sledeći oblik:*

```
Januar 1970.
Po Ut Sr Ce Pe Su Ne
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30 31
```

Zadatak 69 April, 2002. *Napisati program koji učitava sa standardnog ulaza prvo jednu liniju teksta a zatim još jednu liniju sa karakterima koje treba izbaciti iz prve linije. Program treba da izbaci specificirane karaktere iz prve linije i ispiše ono što preostane od iste. Dužina prve linije nije unapred ograničena, tj. za čuvanje te linije treba koristiti listu pri čemu će po jedan karakter biti smješten u svaki element liste. Primjer: ako je unos imao sledeći oblik:*

```
Hello, world!
aeiou,
```

program treba da ispiše:

```
Hll wrld!
```

Zadatak 70 April, 2002. *. Sastaviti program koji ispisuje* `n < 19` *redova Pascalovog trougla koristeći samo 1-dimenzionalni niz i ne koristiti rekurziju. Broj n se zadaje kao jedini sadržaj linije standardnog ulaza. Izveštaj o eventualnim greškama na ulazu ispisati i na standardnom izlazu za poruke o grešci.*

Zadatak 71 April, 2002. *Argumenti komandne linije su imena tri datoteke. Preve dve datoteke u svakom redu sadrže do 80 cifara i u obe datoteke sadržaj je sortiran strogo rastuće po numeričkoj vrednosti broja predstavljenog tim ciframa. Napisati program koji će spojiti te dve datoteke u treću čiji će sadržaj takođe biti sortiran strogo rastuće po numeričkoj vrednosti brojeva koje sadrži.*

Zadatak 72 Jun, 2002. *Neka je* `P = (p1, p2, . . . , pn)` *permutacija brojeva* `1, 2, . . . , n`*. Napisati PASCAL program koji za učitan prirodan broj* `n < 50` *i za učitanu tablicu inverzije ispisuje odgovarajuću permutaciju. Pod tablicom inverzije permutacije* `P` *se podrazumeva niz* `S = (s1, s2, . . . , sn)` *u kom je si jednako broju elemenata permutacije* `P` *koji (u P) stoje levo od broja* `i`*, a veći su od broja* `i`*.*

Zadatak 73 Jun, 2002. *Slika je opisana u kvadratnoj matrici tako da elementi koji određuju sliku popunjeni su cifrom 1, a ostali elementi su popunjeni cifrom 0. kao parametar komandne linije zadaje se ime datoteke u čijoj prvoj liniji se nalazi dimenzija matrice koaj opisuje sliku, a zatim u svakoj liniji nalaze se vrste matrice. Elementi unutar vrste su razdvojeni blankom. napisati program koji u svakoj liniji datoteke REPORT.DAT ispisuje poruke o simetričnosti matrice u odnosu na horizontalnu osu, vertikalnu osu, glavnu dijagonalu, sporednu dijagonalu, centar.*

Zadatak 74 Jun, 2002. *Sprovedena je anketa o popularnosti televizijskih emisija. Broj emisija za koje se glasalo nije veći od 50. Ispitanici su podeljeni u 4 kategorije: mškarci do 30 godina, žene do 30 godina, muškarci stariji od 30 godina, žene starije od 30 godina. Svi su glasali za 3 emisije. Svaka linija u datoteci čije se ime zadaje kao prvo u komandnoj liniji, sadrži podatke o glasanju jednog ispitanikai to sledećim redom: pol ispitanika (m ili z), broj godina, pa zatim šifre emisija za koje je ta osoba glasala. Sifra emisije je niz od najviše 5 karaktera. Napisati program koji u datoteku čije ime se zadaje kao drugo u komandnoj liniji, ispisuje šifre emisija i odgovarajući broj glasova poređane nerstuće po broju osvojenih glasova i to za svaku od kategorija posebno. Emisije za koje se nije glasalo treba preskočiti u ispisivanju.*

Zadatak 75 Jun, 2002. *. Sa standardnog ulaza unosi se najpre jedna linija teksta čija dužina nije ograničena, pa zatim još jedna linija koja sadrži samo karakter koji iz prve linije treba izbaciti. napisati program koji treba da ispiše rezultat odnosno šta je preostalo od linije, pa zatim unošenjem sledećeg karaktera koji se želi izbaciti da ponovi postpak za novodobijenu liniju, i tako dalje po istom principu sve dok se za karakter koji se želi izbaciti ne unese * ili dok od linije ne ostane ništa. Pošto dužina linije nije ograničena, za njeno čuvanje treba koristiti povezanu listu kod koje svaki element čuva po jedno slovo iz linije. Koristiti modularni pristup. Primer jedne sesije bi mogao biti:*

```
programiranje
p
rogramiranje
e
rogramiranj
s
rogramiranj
a
rogrmirnj
*
```

Zadatak 76 Jun, 2002. *Data je datoteka u kojoj se nalazi tekst čiji su naslovi obeleženi etiketom h. Maksimalna dubina naslova je n, gde se ceo broj n zadaje kao argument komandne linije. Svaka etiketa naslova je zatvorena. (Npr.* `<h3>Zadaci za pismeni</h3>`*). Jedini komentari u tekstu sadrže oznake broja strane i oblika su* `<!--xxxx-->` *gde je xxxx najviše četvorocifreni neoznačen broj. Tekst je kodiran bez ikakvih grešaka! Sastaviti program koji iz komandne linije uzima ime gore opisane datoteke i kreira na izlazu datoteku u kojoj se nalazi sadržaj ulaznog teksta. Sadržaj se formira kao niz redova koji sadrže niske obeležene h-etiketama i odgovarajući broj strane. Npr.*

```
ulaz:                          izlaz:
<h3>Zadaci za pismeni</h3>     2.2.3. Zadaci za pismeni.......228
<h4>Pitanja za usmeni</h4>     2.2.3.1. Pitanja za usmeni.....235
```

gde su navedeni naslovi uzastopni. Sadržaj ulazne datoteke se mora formirati pre nego što se u nju upiše.

Zadatak 77 Jun, 2002. *U tekstualnoj datoteci nalaze se podaci o prijemnom ispitu učenika jedne osnovne škole tako što je u svakom redu navedeno: ime i prezime učenika (niz znakova ne duži od 50 znakova), broj poena na osnovu uspeha (decimalan broj), broj poena na prijemnom ispitu iz matematike (decimalan broj) i broj poena na prijemnom ispitu iz maternjeg jezika (decimalan broj). Za učenika koji osvoji manje od 10 poena ukupno na oba prijemna smatra se da nije položio prijemni. Napisati program na C-u koji na osnovu podataka iz ove datoteke formira i prikazuje rang listu učenika. Rang lista sadrži: redni broj, ime i prezime učenika, broj poena na osnovu uspeha, broj poena na prijemnom ispitu iz maternjeg jezika, broj poena na prijemnom ispitu iz matematike i ukupan broj poena i sortirana je opadajuće po ukupnom broju poena. U rang listi se navode prvo oni učenici koji su položili prijemni a potom učenici koji nisu položili prijemni. Između ove dve*

grupe staviti horizontalnu liniju (————————————————————). Ime datoteke navodi se kao argument komandne linije.

Zadatak 78 Jun, 2002. *Napisati program u C-u koji sa standardnog ulaza učitava cifre n i k, a na standardnom izlazu prikazuje najmanji prirodan broj koji počinje cifrom n i ima svojstvo da se smanjuje k puta kada se cifra n premesti sa početka na kraj. Primer: za* `n=3` *i* `k=2` *traženi broj je* `315789473684210526`

Zadatak 79 Jun, 2002. *Svaka linija datoteke čije se ime prosleđuje komandnom linijom sadrži po 6 celih brojeva:* `x1, y1, x2, y2, x3, y3` *koji predstavljaju redom koordinate temena jednog trougla. Linija u datoteci nema više od 100. Napisati program koji uzimajući u obzir samo trouglove koji su jednakostranični, ispituje da li se oni svi mogu "upisati" jedan u drugi (ako je jedan trougao upisan u drugi njegova temena mogu i ne moraju pripadati stranicama ovog drugog). Odgovarajuću poruku štampati na ekran.*

Zadatak 80 Jun, 2002. *Data je datoteka u kojoj se nalazi tekst u kom se nazivi institucija koji se satoje od slova engleske abecede i blanka obeležavaju etiketom name i atributom type.*
Npr. `<name type='institution'>Palata pravde</name>` *maksimalna dužina naziva institucije je n, gde se ceo broj n zadaje kao argument komandne linije. Jedini komentari u tekstu sadrže oznake broja strane i oblika su* `<!- -xxxx- ->` *gde je* `xxxx` *najviše četvorocifreni neoznačen broj. Tekst je kodiran bez ikakvih greški. Sastaviti program koji iz komandne linije uzima ime gore opisane datoteke i kreira na izlazu datoteku index.dat u kojoj se nalazi indeks ulaza koji se formira kao niz redova koji sadrže naziv institucije i broj prve stranice na kojoj se taj naziv pojavio. nazive isntitucija koji se javljaju često (više od m puta, gde se m zadaje kao argument komandne linije) ne unostit u indeks. Program ne trab da pravi razliku između malih i velikih slova.*

Zadatak 81 Septembar, 2002. *Svaki red datoteke čije se ime zadaje komandnom linijom, sadrži po 3 cela broja:* `A, B, C` *(*`A i B` *nisu istovremeno jednaki nuli), koji predstavljaju koeficijente prave u ravni* `Ax+By+C=0`*. Broj redova u datoteci nije veći od 100. Napisati program koji pronalazi i na standardnom izlazu ispisuje sve parove paralelnih pravih, kao i sve trojke pravih koje se seku u jednoj tački. Način prikaza traženih podataka je proizvoljan, ali treba voditi računa o njihovoj preglednosti.*

Zadatak 82 Septembar, 2002. *Grupa od n plesača (na čijim kostimima su redom brojevi od 1 do n) uvežbava svoju plesnu tačku tako što formiraju krug iz kog će redom izlaziti plesači na sledeći način:*

1. *počev od plesača označenog brojem 1, a brojeći udesno (ka plesačima sa većim rednim brojevima), izlazi m-ti plesač*
2. *nakon isključenja, brojanje otpočinje od sledećeg plesača i to u suprotnom smeru, tj. ako se brojalo udesno, počinje se od desnog suseda isključenog plesača i broji se ulevo*
3. *izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni*

Celi brojevi `m, n` *se zadaju kao argumenti komandne linije. Napisati* `C` *program koji ispisuje redne brojeve plesača u redosledu napuštanja kruga.*

Zadatak 83 Septembar, 2002. `N` *osoba obeleženih brojevima* `1, 2, . . . N` *stoji u krugu. Počev od osobe sa rednim brojem 1 broji se* `K` *osoba i K-ta osoba izlazi iz kruga, a potom se nastavlja brojanje preostalih osoba na isti način, počev od prve osobe koja je izašla. Ovo se nastavlja sve dok u krugu ne ostane samo jedna osoba. Napisati program koji sa standardnog ulaza učitava vrednosti za* `N` *i* `K`*, a na standardnom izlazu prikazuje redosled izlaska ljudi iz kruga i redni broj osobe koja poslednja ostaje. Primer: za* `N=4 i K=3` *redosled izlazaka je* `3, 2, 4` *i na kraju ostaje* `1`*.*

Zadatak 84 Septembar, 2002. *Parametar komandne linije je ime datoteke čiji svaki red (izuzev prvog) je oblika* `ime_deteta:ime_roditelja`*. Prvi red sadrži samo ime jednog roditelja čija su sva deca navedena u narednim redovima u već opisanom obliku. Nije obavezno da se sva deca istog roditelja pojavljuju u uzastopnim redovima i nije unapred poznat ukupan broj roditelja. Jednostavnosti radi, može se smatrati: da sve osobe imaju imena sastavljena od slova engleske abecede, da su sva imena međusobno različita (ignorišući razliku malih i velikih slova), da svaki roditelj nema više od četvoro dece i da redovi datoteke nemaju više od 40 karaktera. Napisati program koji za svaku osobu X formira datoteku (čiji je naziv ime osobe) i koja u svakom redu sadrži imena najbližih stričeva, tetki, ujaka osobe* `X` *(misli se na rođenu braću i sestre roditelja osobe* `X`*).*

Zadatak 85 Septembar, 2002. *Slika je opisana u kvadratnoj matrici tako da elementi koji određuju sliku popunjeni su cifrom 1, odnosno cifrom 0. Kao parametar komandne linije zadaje se ime datoteke u čijoj prvoj liniji se nalazi dimenzija matrice koja opisuje sliku, a zatim se u svakoj liniji nalaze vrste matrice. Elementi unutar vrste su razdvojeni blankom. Napisati C program koji, ne koristeći pomoćne matrice, premešta podsliku (čije koordinate gornjeg levog ugla, dužina i širina se zadaju kao argumenti komandne linije) na novu poziciju čiji položaj gornjeg levog ugla se zadaje sa standardnog ulaza. Original i kopija moraju ostati u okvirima polazne matrice. Poruke o eventualnim greškama štampati na standardni izlaz za poruke o grešci.*

Zadatak 86 Septembar, 2002. *Napisati program koji sa standardnog ulaza učitava cifre pozitivnog celog broja (kojih nema više od 100, a na ulazu su jedna pored druge tj. između cifara nema praznih mesta) a na standardnom izlazu ispisuje najmanji pozitivan ceo broj zapisan istim ciframa. Rezultat ne sme počinjati cifrom nula.*

Zadatak 87 Januar, 2002. *Neka su u tekstualnoj datoteci LAVIRINT dati podaci o matrici-lavirintu. Prvi red tekstualne datoteke sadrži broj kolona (80) i broj vrsta (25) a u svakom sledeem redu se nalaze podaci o jednoj vrsti matrice: karakier* `'Z'` *označava da odgovarajue polje matrice predstavlja zid, a karakrer* `'P'` *označava prazan prostor. Napisati program koji na standardnom izlazu prikazuje lavirint učitan iz datoteke ali tako da polja zida prikazuje karakterom* `'X'` *a prazna polja blanko karakterom. Program potom učitava koordinate dve pozicije u lavirintu i utvrđuje da li postoji put kroz lavirint od jedne do druge pozicije (kretanje je mogue samo kroz prazna polja i to u četiri pravca - gore, dole, levo i desno). Ako put postoji program ponovo prikazuje lavirint ali tako da na početnoj poziciji umesto blanko karakrera stoji karakter 'A', na krajnjoj karakter 'B', a na svim ostalim poljima na putu karakrer 'O'. Ako put ne postoji dati odgovarajuu poruku.*

Zadatak 88 Nepoznati rok *Sa standardnog ulaza se unosi ime datoteke čiji prvi red sadrži dimenziju celobrojne kvadratne matrice n* `(n >100)`*, a ostali redovi elemente matrice (vrstu po vrstu). Formirati niz b dimenzije n čiji je prvi član suma elemenata glavne dijagonale, drugi suma elemenata na prvoj donjoj dijagonalinoj paraleli (nju čine elementi odmah ispod glavne dijagonale), treći element suma druge donje dijagonaline paralele, itd. Ispisati niz na standardni izlaz. Sve greške štampati na standardni izlaz za greške.*

Zadatak 89 Nepoznati rok *Sa standardnog ulaza se unose veliki, celi, neoznačeni brojevi sa najviše* `100` *cifara. Ovih brojeva ima manje od 100 ali njihov broj nije unapred poznat. Napisati program koji sabira ovako unete brojeve i na standardni izlaz ispisuje njihov zbir. Napomena: Svaki broj se unosi u posebnom redu a potrebno je voditi računa o korektnosti ulaznih podataka.*