Osnovi programiranja

Beleške sa vežbi
Školska 2005/2006 godine
(drugi semestar)

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

May 17, 2006

Sadržaj

1

Programski jezik C

[1]

1.1 Sortiranje

Niz može biti sortiran ili uređjen u opadajućem, rastućem, neopadajućem i nerastućem poretku. Dato je nekoliko algoritama za sortiranje niza koji se unosi sa ulaza u nerastućem poretku odnosno tako da važi da je `niz[0] >= niz[1] >= ... niz[n]`. Jednostavnom modifikacijom svakim od ovih algoritama niz se može sortirati i u opadajućem, rastućem ili neopadajućem poretku.

Primer 1 *Selection sort*
U prvom prolazu se razmenjuju vrednosti a[0] sa onim članovima ostatka niza koji su veći od njega. Na taj način će se posle prvog prolaza kroz niz a[0] postaviti na najveći element niza.

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

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

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

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

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

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

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

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

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

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

   putchar('\n');

   return 0;

}
```

Primer 2 *Selection sort 2*
Modifikacija prethodnog rešenja radi dobijanja na efikasnosti. Ne vrše se zamene svaki put već samo jednom, kada se pronađe odgovarajući element u nizu sa kojim treba izvršiti zamenu tako da u nizu bude postavljen trenutno najveći element na odgovarajuće mesto.

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

int main()
{

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

     /* Dimenzija niza, indeks najveceg elementa
   u i-tom prolazu,pomocna i brojacke promenljive */
   int n,ind,pom,i,j;

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

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

   /* Unos clanova niza */
```

```
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje - bez stalnih zamena vec se
   pronalazi indeks trenutno najveceg clana niza*/
   for(i=0; i<n-1; i++)
   {
       for(ind=i,j=i+1; j<n; j++)
          if(a[ind]<a[j])
             ind=j;

       /* Vrsi se zamena onda kada na i-tom mestu
        nije najveci element. Tada se na i-to mesto
        postavlja najveci element koji se nalazio na
        mestu ind. */
       if(i != ind)
       {
          pom=a[ind];
          a[ind]=a[i];
          a[i]=pom;
       }
   }
   /* Ispis niza */
   printf("Sortirani niz:\n");
   for(i=0; i<n; i++)
      printf("%d\t",a[i]);

   return 0;

}
```

Primer 3 *bbsort1*
Algoritam sortiranja buble sort poredi dva susedna elementa niza i ako su pogrešno raspoređeni zamenjuje im mesta. Posle poređenja svih susednih parova najmanji od njih će isplivati na kraj niza. Zbog toga se ovaj metod naziva metod mehurića. Da bi se najmanji broj nesortiranog dela niza doveo na svoje mesto treba ponoviti postupak.

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

int main()
{
   /* Dimenzija niza, pomocna promenljiva
   i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   printf("Unsite dimenziju niza\n");
```

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

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

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

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

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

   /* Stampa prazan red */
   putchar('\n');

   /*Regularan zavrsetak rada programa */
   return 0;

}
```

Primer 4 *bbsort2*
Unapredjujemo prethodni algoritam kako bismo obezbedli da se ne vrse provere onda kada je niz već sortiran nego da se u tom slučaju prekine rad.

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

int main()
{
   /* Dimenzija niza, pomocna promenljiva
    i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];
```

```
   /* Promenljiva koja govori da li je izvrsena
   zamena u i-tom prolazu kroz niz pa ako nije
   sortiranje je zavrseno jer su svaka dva
   susedna elementa niza u odgovarajucem poretku */
   int zam;

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

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

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje */
   for(zam=1,i=n-1; zam && i>0; i--)
      for(zam=0,j=0; j<i; j++)
         if(a[j]<a[j+1])
         {
            /* Zamena odgovarajucih clanova niza */
            pom=a[j];
            a[j]=a[j+1];
            a[j+1]=pom;

            /* Posto je u i-tom prolazu
            izvrsena bar ova zamena zam
            se postavlja na 1 sto
            nastavlja sortiranje */
            zam=1;
         }

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

   return 0;

}
```

Primer 5 *isort*
Insert sort, u svakom trenutku je početak niza sortiran a sortiranje se vrši tako što se jedan po jedan element niza sa kraja ubacuje na odgovarajuće mesto.

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

int main()
{
   /* Dimenzija niza, pomocna
   i brojacke promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

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

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

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

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

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

   putchar('\n');

   return 0;

}
```

Primer 6 *Binarno pretrazivanje*

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

```
int main()
{
   /* Dimenzija niza,pomocna i brojacke
      promenljive */
   int n,pom,i,j;

   /* Niz od maksimalno MAXDUZ elemenata*/
   int a[MAXDUZ];

   /* Elemet koji se trazi i pozicija
   na kojoj se nalazi- ukoliko je u nizu*/
   int x,pozicija;

   /* Pomocne promenljive za pretragu */
   int donji, gornji, srednji;

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

   /* Unos clanova niza */
   for(i=0; i<n; i++)
   {
      printf("Unesite %d. clan niza\n",i+1);
      scanf("%d",&a[i]);
   }

   /*Sortiranje*/
   for(i=0; i<n-1; i++)
      for(j=i+1; j<n; j++)
         if(a[i]>a[j])
         {
            pom=a[i];
            a[i]=a[j];
            a[j]=pom;
         }
   /* Unos elementa binarne pretrage */
   printf("Unesite element koji se trazi\n");
   scanf("%d",&x);

   donji = 0;
   gornji = n-1;
   pozicija = -1;

   while(donji<=gornji)
   {
      srednji = (donji + gornji)/2;
      if(a[srednji] == x)
      {
         pozicija = srednji;
         break;
      }
      else
```

```
         if(a[srednji] < x)
            donji = srednji + 1;
         else
            gornji = srednji -1;
   }


   /* Ispis rezultata */
   if(pozicija == -1)
      printf("Trazeni broj se ne nalazi u nizu!\n");
   else
      printf("Broj %d se nalazi na %d poziciji
       sortiranog niza! \n",x,pozicija+1);

   putchar('\n');

   return 0;
}
```

Primer 7 *Sabiranje dva velika broja, njihovo poređenje, unos i ispis, množenje velikog broja cifrom.*

```
#include<stdio.h>
#define MAXDUZ 1000

int unos_broja(int cifre[], int maxduz)
   {
     int brcifara=0;
     char c;

     c=getchar();
     while ( brcifara < maxduz && c >= '0' && c <= '9')
     {
        cifre[brcifara++]=c-'0';
        c=getchar();
     }

     return brcifara;
   }

void obrni(int cifre[],int brcifara)
{
   int i,pom;

   for (i=0; i<brcifara/2; i++)
   {
      pom=cifre[i];
      cifre[i]=cifre[brcifara-i-1];
      cifre[brcifara-i-1]=pom;
   }
}
```

```
void ispisi(int cifre[],int brcifara)
{  int i;
   putchar('\n');
   for (i=brcifara-1; i>=0; i--)
      printf("%d",cifre[i]);
      /* ili
      putchar(cifre[i]+'0');
      */
    putchar('\n');
}

int jednaki(int cifre1[],int cifre2[],
            int brcifara1, int brcifara2)
{
   int i;
   if (brcifara1 != brcifara2) return 0;

   for (i=0; i<brcifara1; i++)
     if (cifre1[i] != cifre2[i]) return 0;
   return 1;
}


int veci(int cifre1[], int brcifara1,
        int cifre2[], int brcifara2)
{
   int i;
   if (brcifara1>brcifara2) return 1;
   if (brcifara1<brcifara2) return 0;

   for (i=brcifara1-1; i>=0; i--)
   {
      if (cifre1[i]<cifre2[i]) return 0;
      if (cifre1[i]>cifre2[i]) return 1;
   }

   return 0;
}


int saberi(int cifre1[], int brcifara1,
           int cifre2[], int brcifara2,
           int cifre[])
{
    int brcifara=0;
    int i,pom,pamtim=0;

    for(i=0; i<brcifara1 || i<brcifara2; i++)
    {
        pom =((i < brcifara1)? cifre1[i] : 0 )
            +((i < brcifara2)? cifre2[i] : 0)
            + pamtim;
```

```
        cifre[i] = pom%10;
        pamtim = pom/10;
    }
    if (pamtim)
    {
       cifre[i]=pamtim;
       brcifara=i+1;
    }
    else brcifara=i;

    return brcifara;
}

int pomnozic(int c,int cifre[],
             int brcifara, int pcifre[])
{
   int pbrcifara=0;
   int i,pamtim=0;
   for (i=0; i<brcifara; i++)
   {
      pcifre[i]=(cifre[i]*c+pamtim)%10;
      pamtim=(cifre[i]*c+pamtim)/10;
   }
   pbrcifara=brcifara;
   if (pamtim)
   {
   pcifre[pbrcifara]=pamtim;
   pbrcifara++;
   }

   return pbrcifara;
}


int main()
{
int d1,d2,d;
int broj1[MAXDUZ], broj2[MAXDUZ], zbir[MAXDUZ];
d1=unos_broja(broj1,MAXDUZ);
d2=unos_broja(broj2,MAXDUZ);

obrni(broj1,d1);
obrni(broj2,d2);
d=saberi(broj1,d1,broj2,d2,zbir);
ispisi(zbir,d);
return 0;
}
```

2

Programski jezik C

[1]

2.1 Rekurzija

C funkcije se mogu rekurzivno koristiti, što znači da funkcija može pozvati samu sebe direktno ili indirektno.

Primer 8 *Štampanje celog broja.*

```
#include<stdio.h>
void printb(long int n)
{
   if(n<0)
   {
      putchar('-');
      n=-n;
   }
   if(n/10)
      printb(n/10);
   putchar(n % 10 + '0');
}

int main()
{
long int b=-1234;
printb(b);
putchar('\n');
return 0;
}
```

Kad funkcija rekurzivno pozove sebe, svakim pozivom pojavljuje se novi skup svih automatskih promenljivih, koji je nezavisan od prethodonog skupa. Prva funkcija printb kao argument dobija broj -12345, ona prenosi 1234 u drugu printb funkciju, koja dalje prenosi 123 u treću, i tako redom do poslednje koja prima 1 kao argument. Ta funkcija štampa 1 i završava sa radom tako da se vraća na prethodni nivo, na kome se štampa dva i tako redom.

Primer 9 *Računanje sume prvih n prirodnih brojeva.*

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

```
#include<stdio.h>
int suma(int n)
{
   if(n!=0)
      return( n + suma(n-1) );
   else return n;
}

main()
{
int S,n;
printf("Unesite n\n");
scanf("%d",&n);
S=suma(n);
printf("S=%d",S);
putchar('\n');
}
```

Primer 10 *Računanje faktorijela prirodnog broja.*

```
#include<stdio.h>
unsigned long fakt(int n)
{
   if(n!=0)
      return( n*fakt(n-1) );
   else return 1;
}

main()
{
int n;
unsigned long f;
printf("Unesite n\n");
scanf("%d",&n);
f=fakt(n);
printf("f=%d",f);
putchar('\n');
}
```

Primer 11 *Fibonačijevi brojevi.*

```
#include<stdio.h>
int fibr(int n)
{
   if((n==1)||(n==2))
      return 1;
   else return(fibr(n-1)+fibr(n-2));
}

int main()
{
int Fn,n;
printf("Unesite n\n");
```

```
scanf("%d",&n);
Fn=fibr(n);
printf("F[%d]=%d",n,Fn);
putchar('\n');
return 0;
}
```

Primer 12 *Iterativna i rekurzivna varijanta računanja sume niza.*

```
int suma_niza_iterativno(int a[], int n)
{
    int suma = 0;
    int i;
    for (i = 0; i<n;  i++)
        suma+=a[i];
    return suma;
}

int suma_niza(int a[], int n)
{
    if (n == 1)
        return a[0];
    else
        return suma_niza(a, n-1)+a[n-1];
}
```

Primer 13 *Stepenovanje prirodnog broja*

```
int stepenuj (int n, int k)
{
    if (k == 0)
        return 1;
    else
        return n*stepenuj(n, k-1);
}
```

2.2 Životni vek i oblast važenja promenjivih, statičke promenljive

Primer 14 *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 15 *Ilustracija statičkih promenljivih.*

```
#include <stdio.h>

void f()
{
    static int a;
    a++;
    printf("%d\n",a);
}

main()
{
int i;
   for (i=0; i<=10; i++)
          f();
}

/* Izlaz iz programa 1 2 3 4 5 6 7 8 9 10 11*/
```

Primer 16 *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);
}
```

2.3 Pokazivači

Pokazivač je promenljiva koja sadrži adresu promenljive.

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

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

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

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

Primer 17 *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
```

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

```
void *pp;
```

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

Primer 18 *Upotreba pokazivača na prazan tip.*

```
#include<stdio.h>

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

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

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

}

/*
 adresa od x je 0012FF78
17 i 0012FF78
 adresa od c je 0012FF74
a i 0012FF74

*/
```

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

2.4 Pokazivači i argumenti funkcija

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

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

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

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

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

a poziv funkcije swap izlgeda sada ovako

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

Primer 19 *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
```

2.5 Zadaci za vežbu

Zadatak 1 *Napisati program koji (a) iterativno (b) rekurzivno računa n-ti Fibonačijev broj, pri čemu se broj n zadaje sa standardnog ulaza. Uporediti brzine izvršavanja ova dva programa za n=5, n=55 i n=95.*

Zadatak 2 *Napisati program u kome se korišćenjem rekurzivne funkcije izračunava NZD brojeva x i y.*

$$nzd(x,y) = \begin{cases} y, & x = 0 \\ nzd(y\%x, x), & x! = 0 \end{cases}$$

Zadatak 3 *Broj je Armstrongov ako je jednak sumi n-tih stepena svojih cifara. Ispitati da li je broj koji se unosi sa standardnog ulaza Armstrongov.*

Zadatak 4 *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.*

Zadatak 5 *(a) Napisati funkciju int palindrom(int broj) koja proverava da li je broj palindrom i vraća vrednost 1 ako jeste, 0 ako nije. Na primer, brojevi 1, 44, 121, 112211, 12321 i 5665 jesu palindromi, a brojevi 123, 67, 8908 nisu.*
(b) Napisati program koji proverava da li je uneti broj palindrom.

Zadatak 6 *Za dati broj može se formirati niz tako da je svaki sledeći član niza dobijen kao suma cifara prethodnog člana niza. Broj je srećan ako se dati niz završava sa jedinicom. Napisati program koji za uneti broj određuje da li je srećan.*

Zadatak 7 *Sa ulaza se unosi broj u osnovi deset i osnova <= 10. Odštampati vrednost datog broja u datoj osnovi.*

Zadatak 8 *Sa ulaza se unosi osnova <= 10 i broj. Proveriti da li je taj broj ispravan broj za datu osnovu i ako jeste izračunati njegovu vrednost u osnovi 10.*

Zadatak 9 *Broj je* **Nivenov** *ako je deljiv sumom svojih cifara.*

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

Zadatak 10 *Napisati program koji izračunava vrednost polinoma u tački x:*

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

3

Programski jezik C

[1]

3.1 Pokazivači i nizovi (polja)

U C-u postoji čvrsta veza između pokazivača i nizova. Bilo koja operacija koja se može ostvariti dopisivanjem indeksa niza može se uraditi i sa pokazivačima.

Deklaracija

`int a[10];`

definiše niz `a` *veličine 10 koji predstavlja blok od 10 uzastopnih objekata nazvanih* `a[0], a[1], ..., a[9]`. *Notacija* `a[i]` *odgovara i-tom elementu niza.*

Ako je `pa` *pokazivač na ceo broj*

`int *pa;`

tada iskaz `pa = &a[0];`

podešava da pa pokaže na nulti element niza `a`*, odnosno* `pa` *sadrži adresu od* `a[0]`.

Ako `pa` *pokazuje na određeni element polja, onda po definiciji* `pa+1` *pokazuje na sledeći element,* `pa+i` *pokazuje na i-ti element posle* `pa`. *Stoga, ako* `pa` *pokazuje na* `a[0]` *tada*

`*(pa+1)`

se odnosi na sadržaj od `a[1]`.

`pa+i` *je adresa od* `a[i]`, *a*

`*(pa+i)`

je sadržaj od `a[i]`.

Ovo sve važi bez obzira na tip ili veličinu elemenata u polju `a`.

Iskaz `pa=&a[0]` *se može napisati kao* ***pa=a jer je ime niza sinonim za lokaciju početnog elementa.***

Primer 20 *Veza između pokazivača i nizova.*

```
#include <stdio.h>

void print_array(int* pa, int n);

main()
{
int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int num_of_elements = sizeof(a)/sizeof(int);
int* pa;
```

[1] Preuzeto sa sajta `http://www.matf.bg.ac.yu/~milena`

```
/* Niz je isto sto i adresa prvog elementa */
printf("Niz a : %p\n", a);
printf("Adresa prvog elementa niza a (&a[0]) : %p\n", &a[0]);
/* Niz a : 0012FF5C
Adresa prvog elementa niza a (&a[0]) : 0012FF5C */

/* Moguce je dodeliti niz pokazivacu odgovarajuceg tipa */
pa = a;

printf("Pokazivac pa ukazuje na adresu : %p\n", pa);
/* Pokazivac pa ukazuje na adresu : 0012FF5C */

/* Nizu nije moguce dodeliti pokazivacku promenljivu
   (nizove mozemo smatrati KONSTANTNIM pokazivacima na prvi element) */
/* a = pa; */

/* Pokazivace je dalje moguce indeksirati kao nizove */
printf("pa[0] = %d\n", pa[0]);
printf("pa[5] = %d\n", pa[5]);
/* pa[0] = 1
   pa[5] = 6 */

/* Medjutim, sizeof(pa) je samo velicina pokazivaca, a ne niza */
printf("sizeof(a) = %d\n", sizeof(a));
printf("sizeof(pa) = %d\n", sizeof(pa));
/* sizeof(a) = 40
   sizeof(pa) = 4 */

/* Pozivamo funkciju za stampanje niza i saljemo joj niz */
print_array(a, num_of_elements);
/* 1 2 3 4 5 6 7 8 9 10 */

/* Pozivamo funkciju za stampanje niza
  i saljemo joj pokazivac na pocetak niza */
print_array(pa, num_of_elements);
/* 1 2 3 4 5 6 7 8 9 10 */
}

/* Prosledjivanje niza u funkciju
    void print_array(int pa[], int n);
   je ekvivalentno prosledjivanju
   pokazivaca u funkciju
    void print_array(int* pa, int n);
   Izmedju ovih konstrukcija nema nikakve razlike.
*/
void print_array(int* pa, int n)
{
    int i;
    for (i = 0; i<n; i++)
        printf("%d ", pa[i]);
    putchar('\n');
```

```
}
```

Prilikom deklaracije treba praviti razliku između niza znakova i pokazivača:

```
char poruka[]="danas je lep dan!";
char *pporuka = "danas je lep dan!";
```

`poruka` *je niz znakova koji sadrži dati tekst. Pojedine znake moguće je promeniti ali se poruka uvek odnosi na isto mesto u memoriji.*
`pporuka` *je pokazivač, koji je inicijalizovan da pokazuje na konstantnu nisku, on može biti preusmeren da pokazuje na nešto drugo, ali rezultat neće biti definisan ako pokušate da modifikujete sadržaj niske (jer je to konstantna niska).*

Ako deklarišemo

```
char *pporuka1 = "danas je lep dan!";
char *pporuka2 = "danas je lep dan!";
char *pporuka3 = "danas pada kisa";
```

tada će pokazivači pporuka1 i pporuka2 pokazivati na isto mesto u memoriji, a pporuka3 na neko drugo mesto u memoriji.

Ako uporedimo (pporuka1==pporuka3) uporediće se vrednosti pokazivača. Ako uporedimo (pporuka1 < pporuka2) uporediće se vrednosti pokazivača. Ako dodelimo pporuka1=pporuka3 tada će pporuka1 dobiti vrednost pokazivača pporuka3 i pokazivaće na isto mesto u memoriji. Neće se izvršiti kopiranje sadržaja memorije!!!

Primer 21 *Vežba pokazivačke aritmetike.*

```
#include <stdio.h>

/* Funkcija pronalazi x u nizu niz
   date dimenzije,
   bez koriscenja indeksiranja.
   Funkcija vraca pokazivac na
   poziciju pronadjenog elementa. */

int* nadjiint(int* niz, int n, int x)
{
    while (n-- >= 0 && *niz!=x)
                niz++;

      return (n>=0)? niz: NULL;
}

main()
{
    int a[]={1,2,3,4,5,6,7,8};
    int* poz=nadjiint(a,sizeof(a)/sizeof(int),4);

    if (poz!=NULL)
        printf("Element pronadjen na poziciji %d\n",poz-a);
}
```

Primer 22

```
int strlen(char *s)
{
```

```
    int n;
    for(n=0; *s != '\0'; s++) n++;
    return n;
}
```

Primer 23

```
/* Funkcija kopira string t u string s */
void copy(char* s, char* t)
{
while (*s++=*t++)
              ;
}

/* Ovo je bio skraceni zapis za sledeci kod
    while(*t != '\0')
        {
        *s=*t;
        s++;
        t++;
        }
    *s = '\0';


*/
```

Primer 24

```
/* 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_compare1(char *s, char *t) {
    /* Petlja tece sve dok ne naidjemo
    na prvi razliciti karakter */
    for (; *s == *t; s++, t++)
        if (*s == '\0') /* Naisli smo na kraj
                           oba stringa, a
                           nismo nasli razliku */
            return 0;


    /* *s i *t su prvi karakteri u kojima
       se niske razlikuju.
       Na osnovu njihovog odnosa,
       odredjuje se odnos stringova */

    return *s - *t;
}

/* Mozemo koristiti i sintaksu kao kod nizova */
int string_compare2(char *s, char *t) {
    int i;
```

```
    for (i = 0; s[i] == t[i]; i++)
        if (s[i] == '\0')
            return 0;
    return s[i] - t[i];
}
```

Primer 25 *Pronalazi prvu poziciju karaktera c u stringu s, i vraća pokazivač na nju, odnosno NULL ukoliko s ne sadrži c.*

```
char* string_char(char *s, char c)
{
    int i;
    for (; *s; s++)
        if (*s == c)
            return s;

    /* Nije nadjeno */
    return NULL;
}
```

Primer 26 *Pronalazi poslednju poziciju karaktera c u stringu s, i vraća pokazivač na nju, odnosno NULL ukoliko s ne sadrži c.*

```
char* string_last_char(char *s, char c)
{
    char *t = s;
    /* Pronalazimo kraj stringa s */
    while (*t++)
        ;

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

    /* Nije nadjeno */
    return NULL;
}
```

Primer 27

```
/* Verzija funkcije strstr implementirane
   bez koriscenja indeksiranja */

#include <stdio.h>

/* proverava da li se niska t nalazi unutar niske s*/
int sadrzi_string(char s[], char t[])
{
    int i;
    for (i = 0; s[i]; i++)
    {
        int j, k;
        for (j=0, k=0; s[i+j]==t[k]; j++, k++)
            if (t[k+1]=='\0')
```

```
                return i;
    }
    return -1;
}


/* proverava da li se niska t nalazi unutar niske s*/
char* sadrzi_string_pok(char* s, char* t)
{
    while(*s)
    {
        char *i, *j;
        for (i = s, j = t; *i == *j; i++,j++)
            if (*(j+1)=='\0')
                return s;
        s++;
    }
    return NULL;
}

/* Cita liniju sa stadnardnog ulaza i
   vraca njenu duzinu */
int getline(char* line, int max)
{
    char *s=line;
    int c;
    while ( max-->0 && (c=getchar())!='\n' && c!=EOF)
        *s++ = c;

    if (c=='\n')
        *s++ = c;

    *s = '\0';
    return s - line;
}


main()
{
    char rec[]="zdravo";
    char linija[100];
    while (getline(linija, 100))
        if (sadrzi_string_pok(linija, rec))
            printf("%s",linija);
}
```

3.2 Zadaci za vežbu

Zadatak 11 *Koristeći pokazivače napisati funkciju koja nadovezuje string t na kraj stringa s. (Pretpostavlja se da u s ima dovoljno prostora.)*

Zadatak 12 januar 2006.(I grupa) *Napisati funkciju koja za celobrojni niz dimenzije* `n`*, proverava da li među elementima niza postoje neka dva koja su jednaka.*

Zadatak 13 januar 2006.(II grupa) *Napisati funkciju koja za dve niske koje se prenose kao parametri utvrđuje da li su anagrami ili ne. Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Na primer, niske "anagram" i "ramgana" jesu anagrami, dok "anagram" i "angrm" nisu.*

Zadatak 14 januar 2006.(II grupa) *Napisati funkciju koja u datom celobrojnom nizu* `A` *dužine* `n` *pronalazi (ako postoji) takav par indeksa* `(i,j)` *da je zbir članova niza sa indeksima od* `i` *do* `j` *jednak zadatom broju* `m`*.*

Zadatak 15 *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 16 *(a) Napisati funkciju koja ispituje da li je jedna reč prefiks druge reči.*
(b) Napisati program koji za svaku liniju teksta koja se unosi sa standardnog ulaza a koja nije duža od 100 karaktera proverava da li je neka reč njen prefiks i štampa odgovarajuću poruku.

Zadatak 17 *(a) Napisati funkciju koja ispituje da li je jedna reč sufiks druge reči.*
(b) Napisati program koji za svaku liniju teksta koja se unosi sa standardnog ulaza a koja nije duža od 100 karaktera proverava da li je neka reč njen sufiks i štampa odgovarajuću poruku.

Zadatak 18 *Učitava se linija po linija teksta. Odštampati svaku od tih linija tako da ima veliko slovo na početku rečenice i sva mala unutar rečenice(., ?, !).*

4

Programski jezik C

[1]

4.1 Strukture

Informacije kojima se opisuje realni svet retko se predstavljaju u elementarnoj formi u vidu celih, realnih, znakovnih konstanti itd. Mnogo češće imamo posla sa složenim objektima koji se sastoje od elemenata raznih tipova. Na primer jednu osobu karakterišu ime, prezime, datum i mesto rođenja.

Struktura predstavlja skup podataka kojim se opisuju neka bitna svojstva objekta. Komponente koje obrazuju strukturu nazivaju se elementi strukture.

Sintaksa strukture:

```
struct ime_strukture
{
tip ime_elementa1;
tip ime_elementa2;
...
}
```

Primer 28 *Primer jednostavne strukture.*

```
struct licnost
{
char ime[31];
char adresa[41];
unsigned starost;
};
```

Sada možemo deklarisati dve osobe na sledeći način:

```
struct licnost osoba1, osoba2;
```

Deklaraciju osobe1 i osobe2 mogli smo da zapišemo i na sledeći način

```
struct licnost
{
char ime[31];
```

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

```
char adresa[41];
unsigned starost;
} osoba1, osoba2;
```

Ukoliko nemamo potrebu da se ličnost koristi dalje u programu mogu se napraviti dve osobe bez davanja imena strukturi:

```
struct
{
char ime[31];
char adresa[41];
unsigned starost;
} osoba1, osoba2;
```

Kada imamo promenljivu strukturnog tipa tada elementima date strukture pristupamo uz pomoc operatora '.'.

Primer 29

```
osoba1.starost=20;
osoba2.starost=21;
...
if (osoba1.starost == osoba2.starost)
     printf(" Osobe su iste starosti");
```

Dozvoljeno je praviti nizove struktura. Npr. niz od 20 elemenata koji sadrži ličnosti:

```
struct licnost nizLicnosti[20];
```

Tada da bi pročitali starost neke ličnosti u nizu pišemo:

```
nizLicnosti[5].starost
```

Može se definisati pokazivač na strukturu.

```
struct licnost *posoba;
```

Tada se pristupanje elementima strukture može vršiti upotrebom operatora '.' na standardni način:

```
(*posoba).ime
(*posoba).adresa
(*posoba).starost
```

ili korišćenjem specijalnog operatora $'->'$ na sledeći način:

```
posoba->ime
posoba->adresa
posoba->starost
```

Primer 30 *Elementi strukture mogu da budu i druge strukture.*

```
struct datum
{
unsigned dan;
unsigned mesec;
unsigned godina;
};
```

```
struct licnost
{
char ime[30];
struct datum datumrodjenja;
};
```

Sada se danu, mesecu i godini datuma rodjenja pristupa na sledeći način:

```
osoba.datumrodjenja.dan = 10;
osoba.datumrodjenja.mesec = 5;
osoba.datumrodjenja.godina = 1986;
```

4.1.1 Operator typedef

Operator typedef omogućava nam da definišemo naša imena za neki od osnovih ili izvedenih tipova. Na primer, možemo da uradimo sledeće:

```
typedef double RealanBroj;
```

Nakon ovoga možemo u tekstu deklarista promenljivu x kao RealanBroj, ona će zapravo biti tipa double.

```
RealanBroj x; /* Umesto: double x;*/
```

Ili, ako želimo da skratimo pisanje za neoznačene duge brojeve tj za unsigned long int to možemo da uradimo na sledeći način

```
typedef unsigned long int VelikiBroj;
```

Sada u kodu možemo da koristimo VelikiBroj kao tip.

Operator typedef je naročito pogodan da bi se izbeglo ponavljalnje reči struct pri deklarisanju strukturnih promenljivih.

```
typedef struct _licnost licnost;
```

Sada deklaracija može da bude:

```
licnost osoba1, osoba2;
/* umesto: struct _licnost osoba1, osoba2; */
```

Kao skraćen zapis za

```
struct _tacka {
    float x;
    float y;
}

typedef struct _tacka tacka;
```

može se koristiti:

```
typedef struct _tacka
{
    float x;
    float y;
} tacka;
```

Primer 31 *Struktura artikal.*

```
typedef struct _artikal
{
    long bar_kod;
    char ime[MAX_IME];
    float pdv;
} artikal;
```

Primer 32 *Program ilustruje osnovne geometrijske algoritme kao i rad sa strukturama.*

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

/* Tacke su predstavljene sa dve koordinate. Strukturom gradimo novi tip podataka. */
typedef struct _tacka
{
    float x;
    float y;
} tacka;

typedef struct _vektor
{
   float x, y;
} vektor;

/* Koordinatni pocetak */
tacka kp={0.0,0.0};

/* Niz tacaka */
tacka niz[100];

/* Pokazivac na strukturu tacke */
tacka *pt;

void IspisiTacku(tacka A)
{
   printf("(%f,%f)\n",A.x,A.y);
}

void IspisiVektor(vektor v)
{
   printf("(%f,%f)\n",v.x,v.y);
}

float duzina(vektor v)
{
   return sqrt(v.x*v.x+v.y*v.y);
}
```

```
vektor NapraviVektor(tacka *pA, tacka *pB)
{
     vektor ab;
     ab.x=pB->x - pA->x;
     ab.y=pB->y - pA->y;
     return ab;
}

float rastojanje(tacka A, tacka B)
{
    float dx=B.x - A.x;
    float dy=B.y - A.y;
    return sqrt(dx*dx+dy*dy);
}

/* Izracunava povrsinu trougla Heronovim obrascem.
   Argumenti funkcije su tri tacke koje predstavljaju temena trougla */
float PovrsinaTrougla(tacka A, tacka B, tacka C)
{
    float a=rastojanje(B,C);
    float b=rastojanje(A,C);
    float c=rastojanje(A,B);

    /* Poluobim. */
    float s=(a+b+c)/2.0;

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


/* Izracunava povsinu konveksnog poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float PovrsinaKonveksnogPoligona(tacka poligon[], int br_temena)
{
int i;
   float povrsina=0.0;

   /* Poligon delimo na trouglove i posebno izracunavamo povrsinu svakoga od njih */
   for (i=1; i<br_temena-1; i++)
      povrsina+=PovrsinaTrougla(poligon[0], poligon[i], poligon[i+1]);

   return povrsina;
}


/* Izracunava obim poligona. Argumenti funkcije su niz tacaka
   koje predstavljaju temena poligona kao i njihov broj */
float Obim(tacka poligon[], int br_temena)
{
    int i;
    float o=0;
```

```
    /* Dodajemo duzine stranica koje spajaju susedna temena */
    for (i=0; i<br_temena-1; i++)
        o+=rastojanje(poligon[i], poligon[i+1]);

    /* Dodajemo duzinu stranice koja spaja prvo i poslednje teme */
    o+=rastojanje(poligon[0], poligon[br_temena-1]);
    return o;
}

main()
{
    tacka poligon[]={{0.0,0.0},
                     {0.0,1.0},
                     {1.0,1.0},
                     {1.0,0.0}};
    printf("Obim poligona je %f\n",Obim(poligon,4));
    printf("Povrsina poligona je %f\n",
              PovrsinaKonveksnogPoligona(poligon,4));
}
```

Primer 33 *Program koji učitava niz studenata i sortira ih po njihovim ocenama.*

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

#define MAX_IME  20

typedef struct _student
{
   char ime[MAX_IME];
   char prezime[MAX_IME];
   int ocena;
} student;

/* Funkcija ucitava rec i vraca njenu duzinu ili
   -1 ukoliko smo dosli do znaka EOF*/
int getword(char word[],int max)
{
    int c, i=0;

    while (isspace(c=getchar()))
         ;

    while(!isspace(c) && c!=EOF && i<max-1)
    {
          word[i++]=c;
          c = getchar();
    }

    word[i]='\0';

    if (c==EOF) return -1;
        else return i;
```

```
}

/* Funkcija ucitava niz studenata, vraca duzinu
   niza koji ucita */
int UcitajPodatke(student studenti[], int max)
{
int i=0;
while(i<max && getword(studenti[i].ime, MAX_IME)>0)
{
if (getword(studenti[i].prezime, MAX_IME) < 0)
        break;
scanf("%d",&studenti[i].ocena);
i++;
}
return i;
}

void IspisiPodatke(student studenti[], int br_studenata)
{
int i;
printf("IME          PREZIME          OCENA\n");
printf("--------------------------------------\n");
for (i=0; i<br_studenata; i++)
    printf("%-20s %-20s %5d\n",studenti[i].
        ime, studenti[i].prezime, studenti[i].ocena);
}

/* Sortiranje studenata po ocenama */
void SelectionSort(student studenti[], int br_studenata)
{
    int i,j;
    for (i=0; i<br_studenata-1; i++)
       for (j=i; j<br_studenata; j++)
           if (studenti[i].ocena<studenti[j].ocena)
           {    student tmp=studenti[i];
                studenti[i]=studenti[j];
                studenti[j]=tmp;
           }
}

main()
{
student studenti[100];
int br_studenata = UcitajPodatke(studenti,100);

SelectionSort(studenti, br_studenata);

IspisiPodatke(studenti, br_studenata);

return 0;
}
```

Primer 34 *Sa standardnog ulaza se učitava niz od* `n (n<100)` *tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka odredjuje konveksni mnogougao i rezultat ispisati na standardni izlaz.*

```
#include<stdio.h>

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

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

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

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

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

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

```
                        mnogougao[i+1],mnogougao[j],mnogougao[j+1]);
        if(i!=0&&i!=n-1&&i+1!=0&&i+1!=n-1)
            konveksan=konveksan && SaIsteStranePrave(mnogougao[i],
                        mnogougao[i+1],mnogougao[0],mnogougao[n-1]);
    }
    for(j=1;konveksan&&j<n-2;j++)
        konveksan=konveksan && SaIsteStranePrave(mnogougao[0],
                        mnogougao[n-1],mnogougao[j],mnogougao[j+1]);

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

}
```

4.2 Rad sa datotekama

Primer 35 *Program demonstrira otvaranje datoteka ("r" - read i "w" - write mod) i osnovne tehnike rada sa datotekama U datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz.*

```
#include <stdio.h>

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

main()
{
        int i;
        int br;

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

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

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

        /* Zatvaramo datoteku */
        fclose(f);

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

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

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

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

        /* Zatvaramo datoteku */
        fclose(f);

}
```

Primer 36 *Program demonstrira "a" - append mod datoteka - nadovezivanje.*

```
#include <stdio.h>

main()
{
        FILE* datoteka;

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

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

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

Primer 37 *Program ilustruje rad sa datotekama. Program kopira datoteku ulaz.txt u datoteku izlaz.txt. Uz svaku liniju se zapisuje i njen broj.*

```
#include <stdio.h>

#define MAX_LINE 256

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

```
int getline(char s[], int lim)
{
        char* c = fgets(s, lim, stdin);
        return c==NULL ? 0 : strlen(s);
}

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

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

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

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

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

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

Primer 38 *Citanje niza struktura iz tektsualne datoteke - artikli prodavnice.*

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

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

```
#include <stdio.h>

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

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

/* Maksimalni broj artikala */
#define MAX_ARTIKALA 1000

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

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

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

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

/* Ispisuje podatke o svim artiklima */
void ispisi_artikle()
{
        int i;
        for (i = 0; i<br_artikala; i++)
                printf("%-5d  %-10s %.2f %.2f    = %.2f\n",
                        artikli[i].bar_kod, artikli[i].ime,
```

```
                        artikli[i].cena, artikli[i].pdv, cena(artikli[i]));
}

main()
{
        FILE* f;


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

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


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

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

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

Primer 39 *Program ilustruje čitanje etiketa iz neke HTML datoteke.*

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

/* Maksimalna duzina etikete */
#define MAX_TAG 100

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

/* Funkcija ucitava sledecu etiketu
   i smesta njen naziv u niz s duzine max.
   Vraca OTVORENA za otvorenu etiketu,
   ZATVORENA za zatvorenu etiketu,
   odnosno GRESKA inace */
int gettag(FILE *f, char s[], int max)
{   int c, i;
    int zatvorenost=OTVORENA;
```

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

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

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

    s[i]='\0';

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

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

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

 FILE* f;


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

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

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

fclose(f);
}
```

4.3 Zadaci za vežbu

Zadatak 19 *Datoteka cije se ime unosi na ulazu sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

Zadatak 20 *Definišemo strukturu VREME na sledeći način:*

```
typedef struct{
    int sat, min, sek;
    } VREME;
```

Sastaviti funkciju sa prototipom `void plus(VREME *t)` *koja povećava za jednu sekundu vreme predstavljano strukturom t.*

Zadatak 21 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.*

5

Programski jezik C

[1]

5.1 Argumenti komandne linije

Primer 40 *Ilustracija rada sa argumentima komandne linije.*

```
/* Program pozivati sa npr.:
        ./a.out
        ./a.out prvi
        ./a.out prvi drugi treci
        ./a.out -a -bc ime.txt
*/

#include <stdio.h>

/* Imena ovih promenljivih mogu biti proizvoljna. Npr.

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

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

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

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

Primer 41 *Program ispisuje opcije navedene u komandnoj liniji.* K&R *rešenje.*

[1] Preuzeto sa sajta `http://www.matf.bg.ac.yu/~milena`

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

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

#include <stdio.h>

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

}

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

Primer 42 *Program ispisuje opcije navedene u komandnoj liniji - jednostavnija verzija.*

```
#include <stdio.h>

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

Primer 43 *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 `+,-,/,*,%`.

```
#include <stdio.h>
unsigned Trampa(unsigned n, int i, int j);

main(int argc, char **argv)
{
    unsigned x; /*broj sa standardnog ulaza ciji se bitovi razmenjuju*/
     int i,j;   /*pozicije bitova za trampu*/

     /*ocitavanje parametara komandne linije i broja sa standarnog ulaza*/
   sscanf(argv[1], "%d", &i);
   sscanf(argv[2], "%d", &j);
   scanf("%u", &x);

   printf("\nNakon trampe vrednost unetog broja je %u\n", Trampa(x,i,j));

}
unsigned Trampa(unsigned n, int i, int j)
{
   //ako se bit na poziciji i razlikuje od bita na poziciji j, treba ih invertovati
   if (  ((n>>i)&1)  !=   ((n>>j)&1)  )   n^= (1<<i) | (1<<j);
   return n;
}
```

Primer 44 *Iz datoteke čije se ime zadaje kao argrument komandne linije, učitati cele brojeve sve dok se ne učita nula, i njihov zbir ispisati u datoteku čije se ime takođe zadaje kao argument komandne linije.*

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

```
        printf("U kojoj datoteci treba ispisati rezultat: ");
        scanf("%s", ime_datoteke_izlaz);
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (izlaz = fopen(ime_datoteke_izlaz, "w")) == NULL)
            printf("Greska : datoteka %s ne moze biti otvorena\n", ime_datoteke_izlaz);
    }

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

5.2 Alokacija memorije

`void* malloc(size_t n)` *vraća pokazivač na n bajtova neinicijalizovane memorije ili* `NULL` *ukoliko zahtev ne može da se ispuni.*

Za njeno korišćenje neophodno je uključiti zaglavlje `stdlib.h`. *Oslobađanje memorije - funkcija* `free`.

Ne sme se koristiti nešto što je već oslobođeno, ne sme se dva puta oslobađati ista memorija.

Primer 45

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

main()
{
int n;
int i;
int *a;

printf("Unesi broj clanova niza : ");
scanf("%d", &n);

/* Kao da ste mogli da uradite
    int a[n];
*/
a = (int*)malloc(n*sizeof(int));

/* Kad god se vrsi alokacija memorije mora se proveriti da li je ona
   uspesno izvrsena!!! */
if (a == NULL)
{
    printf("Nema slobodne memorije\n");
    exit(1);
}

/* Od ovog trenutka a koristim kao obican niz */
```

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

/* Stampamo niz u obrnutom redosledu */
for(i = n-1; i>=0; i--)
    printf("%d",a[i]);

/* Oslobadjamo memoriju*/
free(a);
}
```

Primer 46 *Demonstracija funkcije* `calloc` *- funkcija inicijalizuje sadrzaj memorije na 0.*

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

main()
{
int *m, *c, i, n;

printf("Unesi broj clanova niza : ");
scanf("%d", &n);

/* Niz m NE MORA garantovano da ima sve nule */
m = malloc(n*sizeof(int));
if (m == NULL) {
                printf("Greska prilikom alokacije memorije!\n");
                exit(1);
               }

/* Niz c MORA garantovano da ima sve nule */
c = calloc(n, sizeof(int));
if (c == NULL) {
                printf("Greska prilikom alokacije memorije!\n");
                free(m);
                exit(1);
                }
for (i = 0; i<n; i++)
    printf("m[%d] = %d\n", i, m[i]);

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

free(m);
free(c);
}
```

5.3 Niz pokazivača

Primer 47

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

```
main()
{
/* Niz od tri elemenata tipa int*/
int nizi[3];

/* Niz od tri elemenata tipa int*, dakle
  niz od tri pokazivaca na int*/
int* nizip[3];

/* Alociramo memoriju za prvi element niza*/
nizip[0] = (int*) malloc(sizeof(int));
if (nizip[0] == NULL)
{
    printf("Nema slobodne memorije\n");
    exit(1);
}
/* Upisujemo u prvi element niza broj 5*/
*nizip[0] = 5;
printf("%d", *nizip[0]);

/* Alociramo memoriju za drugi element niza.
   Drugi element niza pokazuje na niz od dva
   elementa*/
nizip[1] = (int*) malloc(2*sizeof(int));
if (nizip[1] == NULL) {
    printf("Nema slobodne memorije\n");
    free(nizip[0]);
    exit(1);
}

/* Pristupamo prvom elementu na koji pokazuje
   pokazivac nizip[1]*/
*(nizip[1]) = 1;

/* Pristupamo sledecem elementu u nizu na koji pokazuje
   nizip[1].
*/
*(nizip[1] + 1) = 2;

printf("%d", nizip[1][1]);

/* Alociramo memoriju za treci element niza nizip. */
nizip[2] = (int*) malloc(sizeof(int));
if (nizip[2] == NULL) {
    printf("Nema slobodne memorije\n");
    free(nizip[0]);
    free(nizip[1]);
    exit(1);
}

*(nizip[2]) = 2;
```

```
printf("%d", *(nizip[2]));

free(nizip[0]);
free(nizip[1]);
free(nizip[2]);
}
```

Primer 48

```
#include <stdio.h>
#include <stdlib.h>
main()
{
/* Niz karaktera*/
char nizc[5];

/* Niz karaktera od cetiri elementa
  ('A', 'n', 'a', '\0')*/
char nizcc[]="Ana";
printf("%s", nizcc);

/* Niz od tri pokazivaca. Prvi pokazuje na
   nisku karaktera Kruska, drugi na nisku karaktera
   Sljiva a treci na Ananas. */
char* nizcp[]={"Kruska", "Sljiva", "Ananas"};

printf("%s", nizcp[0]);
printf("%s", nizcp[1]);
printf("%s", nizcp[2]);
}
```

5.4 Matrice

Primer 49 *Statička alokacija prostora za matricu.*

```
#include <stdio.h>

main()
{
int a[3][3] = {{0, 1, 2}, {10, 11, 12}, {20, 21, 22}};
int i, j;

/* Alternativni unos elemenata matrice
for(i=0; i<3; i++)
    for(j=0; j<3; j++)
        {
        printf("a[%d][%d] = ", i, j);
        scanf("%d", &a[i][j]);
        }
*/

a[1][1] = a[0][0] + a[2][2];
/* a[1][1] = 0 + 22 = 22 */
```

```
printf("%d\n", a[1][1]);   /* 22 */

/* Stampanje elemenata matrice*/
for(i=0; i<3; i++)
    {
    for(j=0; j<3; j++)
        printf("%d\t", a[i][j]);
    printf("\n");
    }
}
```

Nama je potrebno da imamo veću fleksibilnost, tj da se dimenzije matrice mogu uneti kao parametri našeg programa. Zbog toga je neophodno koristiti dinamicku alokaciju memorije.

Primer 50 *Implementacija matrice preko niza.*

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

/* Makro pristupa clanu na poziciji i, j matrice koja ima
   m vrsta i n kolona */
#define a(i,j) a[(i)*n+(j)]

main()
{
   /* Dimenzije matrice */
   int m, n;

   /* Matrica */
   int *a;

   int i,j;

   /* Suma elemenata matrice */
   int s=0;

   /* Unos i alokacija */
   printf("Unesi broj vrsta matrice : ");
   scanf("%d",&m);

   printf("Unesi broj kolona matrice : ");
   scanf("%d",&n);

   a=malloc(m*n*sizeof(int));
   if (a == NULL) {
                    printf("Greska prilikom alokacije memorije!\n");
                    exit(1);
                    }

   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
      {
```

```
      printf("Unesi element na poziciji (%d,%d) : ",i,j);
      scanf("%d",&a(i,j));
      }

   /* Racunamo sumu elemenata matrice */
   for (i=0; i<m; i++)
      for (j=0; j<n; j++)
            s+=a(i,j);

   /* Ispis unete matrice */
   printf("Uneli ste matricu : \n");
   for (i=0; i<m; i++)
   {   for (j=0; j<n; j++)
          printf("%d ",a(i,j));
       printf("\n");
   }

   printf("Suma elemenata matrice je %d\n", s);

   /* Oslobadjamo memoriju */
   free(a);
}
```

5.5 Zadaci za vežbu

Zadatak 22 *Napisati program koji omogućava unos dimenzije kvadratne matrice i unos elemenata matrice sa standardnog ulaza.*

1. *Napisati funkciju koja računa zbir elemenata matrice dimenzija $n \times m$.*
2. *Napisati funkciju koja računa proizvod elemenata ispod glavne dijagonale matrice dimenzija $n \times n$.*

Program treba da odštampa zbir elemenata matrice i proizvod elemenata ispod glavne dijagonale.

6

Programski jezik C

[1]

6.1 Pokazivači na funkcije

Primer 51 *Program demonstrira upotrebu pokazivača na funkcije.*

```
#include <stdio.h>

int kvadrat(int n) { return n*n; }

int kub(int n) { return n*n*n; }

int parni_broj(int n) { return 2*n; }

/* Funkcija izracunava sumu od 1 do n f(i),
   gde je f data funkcija */
int sumiraj(int (*f) (int), int n) {
    int i, suma=0;
    for (i=1; i<=n; i++)
        suma += (*f)(i);

    return suma;
}

main() {
printf("Suma kvadrata brojeva od jedan do 3 je %d\n", sumiraj(kvadrat,3));
printf("Suma kubova brojeva od jedan do 3 je %d\n", sumiraj(kub,3));
printf("Suma prvih pet parnih brojeva je %d\n", sumiraj(parni_broj,5));
} /*Izlaz: Suma kvadrata brojeva od jedan do 3 je 14 Suma kubova
brojeva od jedan do 3 je 36 Suma prvih pet parnih brojeva je 30 */
```

6.2 Matrice - uvežbavanje

Primer 52 *Program ilustruje rad sa kvadratnim matricama i relacijama. Elementi i je u relaciji sa elementom j ako je m[i][j] = 1, a nisu u relaciji ako je m[i][j] = 0.*

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

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

/* Dinamicka matrica je odredjena adresom
   pocetka niza pokazivaca i dimenzijama tj.
   int** a;
   int m,n;
*/

/* Alokacija kvadratne matrice nxn */
int** alociraj(int n)
{
    int** m;
    int i;
    m=malloc(n*sizeof(int*));
    if (m == NULL)
    {
        printf("Greska prilikom alokacije memorije!\n");
        exit(1);
    }

    for (i=0; i<n; i++)
        {
            m[i]=malloc(n*sizeof(int));
            if (m[i] == NULL)
            {
            int k;
            printf("Greska prilikom alokacije memorije!\n");
            for(k=0;k<i;k++)
                    free(m[k]);
            exit(1);
            }
        }

    return m;
}

/* Dealokacija matrice dimenzije nxn */
void obrisi(int** m, int n)
{
    int i;
    for (i=0; i<n; i++)
       free(m[i]);
    free(m);
}

/* Ispis matrice /
void ispisi_matricu(int** m, int n)
{
     int i, j;
      for (i=0; i<n; i++)
      {
```

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

/* Provera da li je relacija predstavljena matricom refleksivna */
int refleksivna(int** m, int n)
{
    int i;
    for (i=0; i<n; i++)
       if (m[i][i]==0)
           return 0;

    return 1;
}

/* Provera da li je relacija predstavljena matricom simetricna */
int simetricna(int** m, int n)
{
    int i,j;
    for (i=0; i<n; i++)
      for (j=i+1; j<n; j++)
         if (m[i][j]!=m[j][i])
             return 0;
    return 1;
}

/* Provera da li je relacija predstavljena matricom tranzitivna*/
int tranzitivna(int** m, int n)
{
    int i,j,k;

    for (i=0; i<n; i++)
       for (j=0; j<n; j++)
           for (k=0; k<n; k++)
               if ((m[i][j]==1)
                   && (m[j][k]==1)
                   && (m[i][k]!=1))
                     return 0;
    return 1;
}

/* Pronalazi najmanju simetricnu relaciju koja sadrzi relaciju a
*/
void simetricno_zatvorenje(int** a, int n)
{
   int i,j;
   for (i=0; i<n; i++)
     for (j=0; j<n; j++)
     {
        if (a[i][j]==1 && a[j][i]==0)
```

```
                    a[j][i]=1;
          if (a[i][j]==0 && a[j][i]==1)
                    a[i][j]=1;
      }
}

main() {
       int **m;
       int n;
       int i,j;

       printf("Unesi dimenziju matrice : ");
       scanf("%d",&n);
       m=alociraj(n);

       for (i=0; i<n; i++)
          for (j=0; j<n; j++)
             scanf("%d",&m[i][j]);

       printf("Uneli ste matricu : \n");

       ispisi_matricu(m,n);

       if (refleksivna(m,n))
            printf("Relacija je refleksivna\n");
       if (simetricna(m,n))
            printf("Relacija je simetricna\n");
       if (tranzitivna(m,n))
            printf("Relacija je tranzitivna\n");

       simetricno_zatvorenje(m,n);

       ispisi_matricu(m,n);

       obrisi(m,n);
}
```

Primer 53 *Izračunati vrednost determinante matrice preko Laplasovog razvoja.*

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

/* Funkcija alocira matricu dimenzije nxn */
int** allocate(int n)
{
     int **m;
     int i;
     m=(int**)malloc(n*sizeof(int*));
     if (m == NULL) {
        printf("Greska prilikom alokacije memorije!\n");
        exit(1);
}
```

```
    for (i=0; i<n; i++)
        {
        m[i]=malloc(n*sizeof(int));
        if (m[i] == NULL)
         {
         int k;
         for(k=0;k<i;k++)
                free(m[k]);
         printf("Greska prilikom alokacije memorije!\n");
         exit(1);
         }
        }

     return m;
}

/* Funkcija vrsi dealociranje date matrice dimenzije n */ void
deallocate(int** m, int n)
{
     int i;
     for (i=0; i<n; i++)
        free(m[i]);
     free(m);
}

/* Funkcija ucitava datu alociranu matricu sa standardnog ulaza */
void ucitaj_matricu(int** matrica, int n)
{
     int i,j;
     for (i=0; i<n; i++)
        for (j=0; j<n; j++)
           scanf("%d",&matrica[i][j]);
}

/* Rekurzivna funkcija koja vrsi Laplasov razvoj */
int determinanta(int** matrica, int n)
{
    int i;
    int** podmatrica;
    int det=0,znak;

    /* Izlaz iz rekurzije je matrica 1x1 */
    if (n==1)
    return matrica[0][0];

    /* Podmatrica ce da sadrzi minore polazne matrice */
    podmatrica=allocate(n-1);
    znak=1;
    for (i=0; i<n; i++)
    {
    int vrsta,kolona;
    for (kolona=0; kolona<i; kolona++)
```

```
        for(vrsta=1; vrsta<n; vrsta++)
            podmatrica[vrsta-1][kolona] = matrica[vrsta][kolona];
    for (kolona=i+1; kolona<n; kolona++)
        for(vrsta=1; vrsta<n; vrsta++)
            podmatrica[vrsta-1][kolona-1] = matrica[vrsta][kolona];

    det+= znak*matrica[0][i]*determinanta(podmatrica,n-1);
    znak*=-1;
    }
    deallocate(podmatrica,n-1);
    return det;
}

main()
{
   int **matrica;
   int n;

   scanf("%d", &n);
   matrica = allocate(n);
   ucitaj_matricu(matrica, n);
   printf("Determinanta je : %d\n",determinanta(matrica,n));
   deallocate(matrica, n);
}
```

6.3 Dinamički niz

Primer 54 *Ilustracija dinamičkog niza.*

```
/* Program za svaku rec unetu sa standardnog
   ulaza ispisuje broj pojavljivanja.
   Verzija sa dinamickim nizom i realokacijom.
*/

#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

/* Rec je opisana imenom i brojem
   pojavljivanja */
typedef struct _rec
{
    char ime[80];
    int br_pojavljivanja;
} rec;

/* Dinamicki niz reci je opisan pokazivacem na
   pocetak, tekucim brojem upisanih elemenata i
   tekucim brojem alociranih elemenata */
rec* niz_reci;
int duzina=0;
```

```
int alocirano=0;

/* Realokacija se vrsi sa datim korakom */
#define KORAK 10

/* Funkcija ucitava rec i vraca njenu duzinu ili
   -1 ukoliko smo dosli do znaka EOF*/
int getword(char word[],int max)
{
    int c, i=0;

    while (isspace(c=getchar()))
         ;

    while(!isspace(c) && c!=EOF && i<max-1)
    {
          word[i++]=c;
          c = getchar();
    }

    word[i]='\0';

    if (c==EOF) return -1;
        else return i;
}

main()
{
char procitana_rec[80];
int i;
while(getword(procitana_rec,80)!=-1)
{
/* Proveravamo da li rec vec postoji u nizu */

   for (i=0; i<duzina; i++)
    /* Ako bi smo uporedili
    procitana_rec == niz_reci[i].ime
    bili bi uporedjeni pokazivaci a ne
    odgovarajuci sadrzaji!!!
    Zato koristimo strcmp. */
    if (strcmp(procitana_rec,
                    niz_reci[i].ime)==0)
    {
    niz_reci[i].br_pojavljivanja++;
    break;
    }

/* Ukoliko rec ne postoji u nizu */
if (i==duzina) {
    rec nova_rec;
    /* Ako bi smo dodelili
        nova_rec.ime = procitana_rec
```

```
        izvrsila bi se dodela pokazivaca
        a ne kopiranje niske procitana_rec
        u nova_rec.ime.
        Zato koristimo strcpy!!! */
    strcpy(nova_rec.ime,procitana_rec);
    nova_rec.br_pojavljivanja=1;

    /* Ukoliko je niz "kompletno popunjen"
        vrsimo realokaciju */
    if (duzina==alocirano)
    {
    alocirano+=KORAK;

    /* Sledeca linija zamenjuje blok
        koji sledi i moze se
        koristiti alternativno. Blok je
        ostavljen samo da bi
        demonstrirao korisnu tehniku */
    /*
    niz_reci=realloc(niz_reci,
                    (alocirano)*sizeof(rec)); */

    {
    /* alociramo novi niz, veci
        nego sto je bio prethodni */
    rec* novi_niz=(rec *)malloc(alocirano*sizeof(rec));

    /* Kopiramo elemente starog niza u novi */
    for (i=0; i<duzina; i++)
            novi_niz[i]=niz_reci[i];
    /* Uklanjamo stari niz */
    free(niz_reci);
    /* Stari niz postaje novi */
    niz_reci=novi_niz;
    }

    if (niz_reci==NULL)
    {
    printf("Greska prilikom
                    alokacije memorije");
    exit(1);
    }
    }
    /* Upisujemo rec u niz */
    niz_reci[duzina]=nova_rec;
    duzina++;
} }

/* Ispisujemo elemente niza */ for(i=0; i<duzina; i++)
    printf("%s - %d\n",niz_reci[i].ime,
                niz_reci[i].br_pojavljivanja);
```

```
free(niz_reci); }
```

6.4 Zadaci za vežbu

Zadatak 23 *Ispisati na izlaz sumu celih brojeva koji se unose kao argumenti komandne linije.*

Zadatak 24 *Napisati funkciju koja omogućava računanje proizvoda dve kvadratne matrice dimenzija $n \times n$. Napisati program koji omogućava unošenje dve kvadratne matrice i štampanje proizvoda te dve matrice.*

Zadatak 25 Jun, 2004. *Napisati funkciju koja računa multiplikativnu otpornost datog pozitivnog broja. Multiplikativna otpornost se računa na sledeći način $n_0 = n$, n_k je jednak proizvodu cifara broja n_{k-1}, $k = 1, 2...$, multiplikativna otpornost je najmanje k za koje je n_k jednocifren broj. Napisati program koji iz datoteke čije se ime zadaje na ulazu čita brojeve, gde su brojevi zapisani po jedan u svakom redu i u drugu datoteku čije se ime zadaje takođe na ulazu upisuje red po red date brojeve i njihovu multiplikativnu otpornost.*

7

Programski jezik C

[1]

7.1 qsort

Primer 55 *Implementacija funkcije qsort.*

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

void printarray(int v[], int left, int right)
{
    int i;
    for (i=left; i<=right; i++)
            printf("%d  ",v[i]);
    putchar('\n');
}

void swap(int v[], int i, int j)
{
    int tmp=v[i];
    v[i]=v[j];
    v[j]=tmp;
}

/* qsort:  sortira v[left]...v[right] u rastucem poretku */
void qsort(int v[], int left, int right)
{
    int i, last;

    /* ne radi nista ako niz sadrzi  */
    /* manje od dva elementa */
    if (left >= right)
        return;
    /* prebaci element particioniranja (pivot)*/
    /* u v[left] */
```

[1] Preuzeto sa sajta `http://www.matf.bg.ac.yu/~milena`

```
    swap(v, left, (left + right)/2);
    last = left;

    /* partition */
    for (i = left + 1; i <= right; i++)
        if (v[i] < v[left])
            swap(v, ++last, i);

    /* restore partition  elem */
    swap(v, left, last);

    /* Sortiraj preostala dva dela niza */
    qsort(v, left, last-1);
    qsort(v, last+1, right);
}

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

    printarray(array, 0, n-1);
    qsort(array, 0, n-1);
    printarray(array, 0, n-1);

}
```

7.2 Sortiranje — generička funkcija

Sortiranje niza celih brojeva (jedan od algoritama)

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

Sortiranje iz programa mozemo da izdvojimo u funkciju:

```
void sort_int(int a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(a[i]<a[j])
        {
        int pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje niza realnih brojeva:

```
void sort_float(float a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(a[i]<a[j])
        {
        float pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Razlike:

- *prvi argument funkcije;*
- *pomoćna promenljiva;*
- *poređenje.*

Sortiranje studenata po oceni ukoliko je data struktura student:

```
typedef struct _student {
   char ime[MAX_IME];
   char prezime[MAX_IME];
   int ocena;
} student;

void sort_po_oceni(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(a[i].ocena < a[j].ocena)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje studenta po prezimenu:

```
void sort_po_prezimenu(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(strcmp(a[i].prezime, a[j].prezime)<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sortiranje studenta po imenu:

```
void sort_po_imenu(student a[], int n)
{
  for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if(strcmp(a[i].ime, a[j].ime)<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Kako da napravimo jednu funkciju koja sortira studente bez obzira na kriterijum?
Prvo moramo da izdvojimo funkciju poređenja:

```
int poredi_po_oceni(student st1, student st2)
{
return st1.ocena - st2.ocena;
}

int poredi_po_prezimenu(student st1, student st2)
{
return strcmp(st1.prezime, s2.prezime);
}

int poredi_po_imenu(student st1, student st2)
{
return strcmp(st1.ime, st2.ime);
}
```

Funkcija poređenja vraća 0 ukoliko su elementi jednaki, broj manji od nule ukoliko je prvi manji od drugog i broj veći od nule ukolikoje prvi veći od drugog.

```
void sort_po_imenu(student a[], int n)
{
for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        /*if(poredi_po_prezimenu(a[i], a[j])<0)*/
        /*if(poredi_po_oceni(a[i], a[j])<0)*/
        if(poredi_po_imenu(a[i], a[j])<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Sada možemo da dodamo još jedan argument funkciji sortiranja i tako da dobijemo jednu funkciju umesto tri:

```
void sort_studente(student a[], int n,
                     int (*f)(student, student))
```

```
{
for(i=0; i<n-1; i++)
    for(j=i+1; j<n; j++)
        if((*f)(a[i], a[j])<0)
        {
        student pom=a[i];
        a[i]=a[j];
        a[j]=pom;
        }
}
```

Šta dalje? Kako da dobijemo jednu funkciju sortiranja bez obzira na tip elemenata niza? Teba da rešimo sledeće stvari:

- *razmena mesta elemenata ne sme da zavisi od tipa elemenata koji se razmenjuju.*
- *potpis funkcije poređenja ne sme da zavisi od tipa elemenata koji se porede kako bi on bio jedinstven.*
- *prvi argument funkcije ne sme da zavisi od tipa elemenata niza.*

Da bi smo razmenili dva elementa potrebna nam je pomoćna promenljiva u kojoj privremeno čuvamo neku vrednost. Ako ne znamo tip elementa onda ne možemo da napravimo pomoćnu promenljivu. Ali zato mozemo da koristeći funkciju malloc odvojimo neko mesto u memoriji za smestanje elementa koji nam u datoj situaciji treba. Koliko je to mesto? Nekada 4 bajta, npr za int, a nekada dosta veće, npr za studenta. Kako funkcija sortiranja zna koliko mesta treba da odvoji? Znace tako sto ćemo joj tu veličinu proslediti kao argument. Sada, dakle umesto pomoćne promenljive, imamo blok u memoriji, a umesto naredbe dodele koristićemo funkciju `memcpy` *koja kopira deo memorije sa jednog mesta na drugo mesto.*

Dakle, razmenu ćemo da radimo na sledeći način:

```
void* tmp = malloc(size);
memcpy(tmp, adresa_itog, size);
memcpy(adresa_itog, adresa_jtog, size);
memcpy(adresa_jtog, tmp, size);
free(tmp);
```

Potpis funkcije poređenja ne sme da zavisi od tipa elemenata koji se porede. To se može postići koristeći pokazivač na tip void.

Na primer, poređenje dva cela broja:

```
int poredi_br(void* a, void* b) {
    int br_a = *(int*)a;
    int br_b = *(int*)b;

    return br_a-br_b;
}
```

Na primer, poređenje dva realna broja:

```
int poredi_br(void* a, void* b) {
    float br_a = *(float*)a;
    float br_b = *(float*)b;

    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
}
```

Na primer, poređenje dva studenta po oceni

```
int poredi_br(void* a, void* b) {
    student student1 = *(studnet*)a;
    student student2 = *(studnet*)b;

    return student1.ocena-student2.ocena;
}
```

Sada funkcija poređenja ima uvek potpis

```
int poredi(void* a, void* b)
```

i može se kao parametar proslediti našoj funkciji sortiranja.

Primer 56
```
/* Genericka funkcija sortiranja -
   nezavisna od tipa elemenata niza
   koji se sortira */
#include <stdlib.h>

void sort(void* a, int n, int size,
             int (*poredi)(void*, void*))
{
  int i, j;
  for (i = 0; i<n-1; i++)
    for (j = i+1; j<n; j++)
    {
        void* adresa_itog = (char*)a+i*size;
        void* adresa_jtog = (char*)a+j*size;

        if (poredi(adresa_itog, adresa_jtog)<0)
        {
            void* tmp = malloc(size);
            memcpy(tmp, adresa_itog, size);
            memcpy(adresa_itog, adresa_jtog, size);
            memcpy(adresa_jtog, tmp, size);
            free(tmp);
        }

    }
}


int poredi_br(void* a, void* b) {
    int br_a = *(int*)a;
    int br_b = *(int*)b;

    return br_a-br_b;
}

int poredi_float(void* a, void* b) {
    float br_a = *(float*)a;
    float br_b = *(float*)b;
```

```
    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
}

main() {
    int a[] = {8, 2, 1, 9, 3, 7, 6, 4, 5};
    float b[] = {0.3, 2, 5, 5.8, 8}
    int n = sizeof(a)/sizeof(int);
    int nf = sizeof(b)/sizeof(float);
    int i;

    sort(a, n, sizeof(int), &poredi_br);

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

    sort(b, nf, sizeof(float), &poredi_float);

    for (i = 0; i < n; i++)
        printf("%f ", b[i]);
    putchar('\n');
}
```

7.3 qSort funkcija iz standardne biblioteke

Primer 57 *qSort-Upotreba.*

```
/* Ilustracija upotrebe funkcije qsort iz stdlib.h
   Sortira se niz celih brojeva.
*/

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

/* const znaci da ono na sta pokazuje a (odnosno b)
   nece biti menjano u funkciji */
int poredi(const void* a, const void* b)
{
    return *((int*)a)-*((int*)b);
}

int poredi_float(const void* a, const void* b)
{
    float br_a = *(float*)a;
    float br_b = *(float*)b;

    if (br_a > br_b) return 1;
    else if (br_a < br_b) return -1;
    else return 0;
```

```
}


main()
{
    int i;
    int niz[]={3,8,7,1,2,3,5,6,9};
    float nizf[]={3.0,8.7,7.8,1.9,2.1,3.3,6.6,9.9};

    int n=sizeof(niz)/sizeof(int);
    qsort((void*)niz, n, sizeof(int),&poredi);
    for(i=0; i<n; i++)
        printf("%d",niz[i]);


    n=sizeof(nizf)/sizeof(float);
    qsort((void*)nizf, n, sizeof(float),&poredi_float);
    for(i=0; i<n; i++)
            printf("%f",nizf[i]);

}
```

Primer 58 *Binarno pretraživanje - korišćenje ugrađene bsearch funkcije.*

```
/* Funkcija ilustruje koriscenje ugradjene funkcije bsearch */
#include <stdlib.h>

int poredi(const void* a, const void *b)
{
    return *(int*)a-*(int*)b;
}

main()
{
    int x=-1;
    int niz[]={1,2,3,4,5,6,7,8,9,10,11,12};

    int* elem=(int*)bsearch((void*)&x,(void*)niz,
        sizeof(niz)/sizeof(int),sizeof(int),&poredi);

if (elem==NULL)
    printf("Element nije pronadjen\n");
else
    printf("Element postoji
                na poziciji %d\n",elem-niz);
}
```

7.4 Generičko sortiranje reči

Primer 59 *Sortiranje reči. Ako se sortira niz stringova, onda svaki element je sam po sebi pokazivač tipa* `char *`*, te funkcija poređenja tada prima podatke tipa* `char **` *koji se konvertuju u svoj tip i derefenciraju radi dobijanja podataka tipa* `char *`*.*

```
/* Ilustracija upotrebe funkcije qsort iz stdlib.h
   Sortira se niz reci i to ili leksikografski
   ili po duzini
*/

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

int poredi(const void* a, const void* b)
{
    char *s1 = *(char **)a;
    char *s2 = *(char **) b;
    return strcmp(s1, s2);

    /* Prethodno je ekvivalentno sa:
    return strcmp(*(char**)a,*(char**)b); */
}

int poredi_po_duzini(const void* a, const void* b)
{
    char *s1 = *(char **) a;
    char *s2 = *(char **) b;
    return strlen(s1) - strlen(s2);
    /* Prethodno je ekvivalentno sa:
    return strlen(*(char**)b)-strlen(*(char**)a); */
}

main()
{
    int i;
    char* nizreci[]= {"Jabuka","Kruska","Sljiva","Dinja","Lubenica"};

    qsort((void*)nizreci,5,
            sizeof(char*),&poredi_po_duzini);

    for (i=0; i<5; i++)
        printf("%s\n",nizreci[i]);

    qsort((void*)nizreci,5,
            sizeof(char*),&poredi);

    for (i=0; i<5; i++)
         printf("%s\n",nizreci[i]);
}
```

7.5 Zadaci za vežbu:

Zadatak 26 *Napisati program koji sa standardnog ulaza ucitava 2 stringa, s i t (duzine ¡=20), sortira nizove njihovih karaktera (biblioteckom qsort funkcijom) i ispituje i stampa da li su s i t*

anagrami (npr. vrata, vatra).

Zadatak 27 *Napisati program koji sa standardnog ulaza ucitava prvo ceo broj n (n¡=10) a zatim niz S od n stringova (maksimalna duzina stringa je 20), sortira niz S (biblioteckom funkcijom qsort) i proverava da li u njemu ima identicnih stringova.*

Zadatak 28 *Napisati program u kome se prvo inicijalizuje staticki niz struktura osoba sa clanovima ime i prezime(uredjen u rastucem poretku prezimena) sa ¡=10 elemenata, a zatim se ucitava jedan karakter i pronalazi(sa bsearch) i stampa jedna struktura iz niza osoba cije prezime pocinje tim kakrakterom(ako takva postoji).*

8

Programski jezik C

[1]

8.1 Liste

Primer 60 *Ubacivanje na početak jednostruko povezane liste - verzija sa **. Ispis i oslobađanje liste realizovani iterativno.*

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

typedef struct cvor
{
    int br;
    struct cvor* sl;
} CVOR;

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */

CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}

/* Zbog prenosa po vrednosti, sledeca funkcija ne radi ispravno */
/*
```

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

```
void ubaci_na_pocetak(CVOR* l, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = l;
    l = novi;  /* Ovde se menja lokalna kopija pokazivaca l, a
              ne l iz funkcije pozivaoca (main) */
}
*/


/* Ubacuje dati broj na pocetak liste.
   Pokazivac na pocetak liste se prenosi preko pokazivaca, umesto po
   vrednosti, kako bi mogla da mu se izmeni vrednost. */
void ubaci_na_pocetak(CVOR** pl, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = *pl;
    *pl = novi;
}

/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Sledeca funkcija je neispravna */
/*
void oslobodi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t!=NULL; t = t->sl)
        free(t);
        /* Ovde se unistava sadrzaj cvora na koji ukazuje t.
           Korak petlje t = t->sl nece moci da se izvrsi */

}
*/

/* Oslobadjanje liste : iterativna verzija */
void oslobodi_listu(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}
```

```
main()
{
    CVOR* l = NULL;
    int i;
    for (i = 0; i<10; i++)
        ubaci_na_pocetak(&l, i);

    ispisi_listu(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

Primer 61 *Ubacivanje na početak jednostruko povezane liste - verzija sa eksplicitnim vraćanjem novog početka liste. Ispis i oslobađanje liste su realizovani rekurzivno.*

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

typedef struct cvor
{
    int br;
    struct cvor* sl;
} CVOR;

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */

CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}

/* Ubacuje dati broj na pocetak date liste.
   Funkcija pozivaocu eksplicitno vraca pocetak rezultujuce liste.*/
CVOR* ubaci_na_pocetak(CVOR* l, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = l;
    return novi;
}
```

```
/* Ispisivanje liste : rekurzivna verzija */
void ispisi_listu(CVOR* l)
{
    if (l != NULL)
    {
        printf("%d ", l->br);
        ispisi_listu(l->sl);
    }
}


/* Ispisivanje liste unatrag : rekurzivna verzija */
/* Prethodna funkcija se lako modifikuje tako da ispisuje listu unazad */
void ispisi_listu_unazad(CVOR* l)
{
    if (l != NULL)
    {
        ispisi_listu_unazad(l->sl);
        printf("%d ", l->br);
    }
}

/* Oslobadjanje liste : rekurzivna verzija */
void oslobodi_listu(CVOR* l)
{
    if (l != NULL)
    {
        oslobodi_listu(l->sl);
        /* Prvo se oslobadja poslednji element liste */
        /* printf("Oslobadjam %d\n", l->br); */
        free(l);
    }
}

main()
{
    CVOR* l = NULL;
    int i;
    for (i = 0; i<10; i++)
        l = ubaci_na_pocetak(l, i);

    ispisi_listu(l);
    putchar('\n');

    ispisi_listu_unazad(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

Primer 62 *Ubacivanje na kraj jednostruko povezane liste - verzija sa ** - iterativna i rekurzivna verzija*

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

typedef struct cvor
{
    int br;
    struct cvor* sl;
} CVOR;

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */

CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}

/* Ubacuje dati broj na pocetak liste.
   Pokazivac na pocetak liste se prenosi preko pokazivaca, umesto po
   vrednosti, kako bi mogla da mu se izmeni vrednost.
   Iterativna verzija funkcije */
/* Ubacivanje na kraj liste je neefikasna operacija */
void ubaci_na_kraj(CVOR** pl, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = 0;

    if (*pl == NULL)
        *pl = novi;
    else
    {
        /* Pronalazimo poslednji element liste - t*/
        CVOR* t;
        for (t=*pl; t->sl!=NULL; t=t->sl)
            ;
        t->sl = novi;
    }

}

/* Rekurzivna varijanta prethodne funkcije */
void ubaci_na_kraj_rekurzivno(CVOR** pl, int br)
{
```

```
    if (*pl == NULL)
    {
        CVOR* novi = napravi_cvor(br);
        *pl = novi;
    }
    else
        ubaci_na_kraj_rekurzivno( &((*pl)->sl) ,br);
}

/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Iterativna verzija funkcije koja oslobadja listu */
void oslobodi_listu(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

main()
{
    CVOR* l = NULL;
    int i;
    for (i = 0; i<5; i++)
        ubaci_na_kraj(&l, i);
    for (; i<10; i++)
        ubaci_na_kraj_rekurzivno(&l, i);

    ispisi_listu(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

Primer 63 *Ubacivanje na kraj jednostruko povezane liste - verzija sa eksplicitnim vraćanjem nove liste - iterativna i rekurzivna verzija.*

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

typedef struct cvor
{
    int br;
```

```
    struct cvor* sl;
} CVOR;

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */

CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}

/* Funkcija vraca pocetak rezultujuce liste */
CVOR* ubaci_na_kraj(CVOR* l, int br)
{
    CVOR* novi = napravi_cvor(br);
    novi->sl = NULL;

    if (l == NULL)
        return novi;
    else
    {
        CVOR* t;
        for (t = l; t->sl!=NULL; t=t->sl)
            ;
        t->sl = novi;

        /* Pocetak se nije promenio */
        return l;
    }
}

/* Rekurzivna varijanta prethodne funkcije.
   I ova funkcija vraca pokazivac na pocetak rezultujuce liste  */
CVOR* ubaci_na_kraj_rekurzivno(CVOR* l, int br)
{
    if (l == NULL)
    {
        CVOR* novi = napravi_cvor(br);
        return novi;
    }

    l->sl = ubaci_na_kraj_rekurzivno(l->sl, br);
    return l;
```

```
}

/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Iterativna verzija funkcije koja oslobadja listu */
void oslobodi_listu(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

main()
{
    CVOR* l = NULL;
    int i;
    for (i = 0; i<5; i++)
        l = ubaci_na_kraj(l, i);
    for (; i<10; i++)
        l = ubaci_na_kraj_rekurzivno(l, i);

    ispisi_listu(l);
    putchar('\n');

    oslobodi_listu(l);
}
```

Primer 64 *Ubacivanje na odgovarajuće mesto sortirane jednostruko povezane liste - verzija sa ** - iterativna i rekurzivna verzija*

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

typedef struct cvor
{
    int br;
    struct cvor* sl;
} CVOR;

/* Pomocna funkcija koja kreira cvor liste sa datim sadrzajem.
   Funkcija kreira cvor i postavlja mu sadrzaj na dati broj.
   Polje sl ostaje nedefinisano.
   Funkcija vraca pokazivac na kreirani cvor. */
```

```
CVOR* napravi_cvor(int br)
{
    CVOR* novi = (CVOR*)malloc(sizeof(CVOR));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    novi->br = br;
    return novi;
}
/* Kljucna ideja u realizaciji ove funkcije je pronalazenje poslednjeg
   elementa liste ciji je kljuc manji od datog elementa br.
*/
void ubaci_sortirano(CVOR** pl, int br)
{
    CVOR* novi = napravi_cvor(br);

    /* U sledeca dva slucaja ne postoji cvor ciji je kljuc manji
       od datog broja (br)
       - Prvi je slucaj prazne liste
       - Drugi je slucaj kada je br manji od prvog elementa

       U oba slucaja ubacujemo na pocetak liste.
    */
    if (*pl == NULL || br < (*pl)->br)
    {
        novi->sl = *pl;
        *pl = novi;
        return;
    }

    /* Krecemo od pocetka i idemo dalje sve dok t nije poslednji
       manji element liste ili eventualno bas poslednji */
    CVOR* t;
    for (t = *pl; t->sl!=NULL && t->sl->br < br; t=t->sl)
        ;
    novi->sl = t->sl;
    t->sl = novi;
}

/* Rekurzivna verzija prethodne funkcije */
void ubaci_sortirano_rekurzivno(CVOR** pl, int br)
{
    if (*pl == NULL || br < (*pl)->br)
    {
        CVOR* novi = napravi_cvor(br);
        novi->sl = *pl;
        *pl = novi;
        return;
    }
```

```
    ubaci_sortirano(&((*pl)->sl), br);
}

/* Ispisivanje liste : iterativna verzija */
void ispisi_listu(CVOR* l)
{
    CVOR* t;
    for (t = l; t != NULL; t=t->sl)
        printf("%d ", t->br);
}


/* Iterativna verzija funkcije koja oslobadja listu */
void oslobodi_listu(CVOR* l)
{
    while (l)
    {
        CVOR* tmp = l->sl;
        free(l);
        l = tmp;
    }
}

main()
{
    CVOR* l = NULL;
    CVOR* k = NULL;
    int i;

    ubaci_sortirano(&l, 5);
    ubaci_sortirano(&l, 8);
    ubaci_sortirano(&l, 7);
    ubaci_sortirano(&l, 6);
    ubaci_sortirano(&l, 4);

    ubaci_sortirano_rekurzivno(&k, 5);
    ubaci_sortirano_rekurzivno(&k, 8);
    ubaci_sortirano_rekurzivno(&k, 7);
    ubaci_sortirano_rekurzivno(&k, 6);
    ubaci_sortirano_rekurzivno(&k, 4);

    ispisi_listu(l);
    putchar('\n');

    ispisi_listu(k);
    putchar('\n');

    oslobodi_listu(l);
}
```

8.1.1 Dvosturko povezana kružna lista

Primer 65 *Napisati funkciju koja omogućava umetanje čvora u dvostruko povezanu kružnu listu kao i izbacivanje čora iz dvostruko povezane kružne liste. Omogućiti i štampanje podataka koje čuva lista.*

```
/* Program implementira deciju razbrajalicu eci-peci-pec i sluzi
da ilustruje rad sa dvostruko povezanim kruznim listama */

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

/* Dvostruko povezana lista */
typedef struct _cvor
{
   int broj;
   struct _cvor* prethodni, *sledeci;
} cvor;

/* Umetanje u dvostruko povezanu listu */
cvor* ubaci(int br, cvor* lista)
{
    cvor* novi=(cvor*)malloc(sizeof(cvor));
    if (novi==NULL)
    {   printf("Greska prilikom alokacije memorije \n");
        exit(1);
    }
    novi->broj=br;

    if (lista==NULL)
    {
       novi->sledeci=novi;
       novi->prethodni=novi;
       return novi;
    }
    else
    {
       novi->prethodni=lista;
       novi->sledeci=lista->sledeci;
       lista->sledeci->prethodni=novi;
       lista->sledeci=novi;
       return novi;
    }
}

/* Ispis liste */
void ispisi(cvor* lista)
{
    if (lista!=NULL)
    {   cvor* tekuci=lista;
        do
        {   printf("%d\n",tekuci->broj);
            tekuci=tekuci->sledeci;
```

```
        } while (tekuci!=lista);
    }
}

/* Izbacivanje datog cvora iz liste */
cvor* izbaci(cvor* lista)
{
    if (lista!=NULL)
    {   cvor* sledeci=lista->sledeci;
        if (lista==lista->sledeci)
        {   printf("Pobednik %d\n",lista->broj);
            free(lista);
            return NULL;
        }

        printf("Ispada %d\n",lista->broj);

        lista->sledeci->prethodni=lista->prethodni;
        lista->prethodni->sledeci=lista->sledeci;
        free(lista);
        return sledeci;
    }
    else return NULL;
}

main()
{
    /* Umecemo petoro dece u listu */
    cvor* lista = NULL;
    lista=ubaci(1,lista);
    lista=ubaci(2,lista);
    lista=ubaci(3,lista);
    lista=ubaci(4,lista);
    lista=ubaci(5,lista);
    lista=lista->sledeci;

    int smer = 0;
    /* Dok ima dece u listi */
    while(lista!=NULL)
    {   int i;

    /* brojimo 13 slogova u krug i
           u svakom brojanju menjamo smer obilaska*/
        for (i=1; i<=13; i++)
           lista = 1-smer ? lista->sledeci : lista->prethodni;

        lista=izbaci(lista);
        smer = smer ? 0 : 1;
    }
    ispisi(lista);
}
```

8.2 Zadaci za vežbu

Zadatak 29 *Brojeve sa ulaza smeštati u listu sve dok se ne unese nula, a zatim dobijenu listu ispisati na izlaz.*

1. *Zadatak realizovati dodavanjem elemenata liste na početak liste.*
2. *Zadatak realizovati tako da listu koja se formira bude sortirana.*
3. *Zadatak realizovati dodavanjem elemenata liste na kraj liste a listu ispisati unazad.*

Zadatak 30 Jun, 2004. Igrupa *Data je datotka* `brojevi.txt` *koja sadrži cele brojeve, po jedan u svakom redu.*

1. *Napisati funkciju koja iz zadate datoteke učitava brojeve i smešta ih u listu.*
2. *Napisati funkciju koja u jednom prolazu kroz zadatu listu celih brojeva pronalazi maximalan strogo rastući podniz.*
3. *Koristeći funkcije pod a) i b) napisati program koji u datoteku* `Rezultat.txt` *upisuje nađeni strogo rastući podniz.*

Zadatak 31 *Grupa od n plesača (na čijim kostimima su u smeru kazaljke na satu redom brojevi od* 1 *do n) izvodi svoju plesnu tačku tako što formiraju krug iz kog najpre izlazi k-ti plesač (odbrojava se počev od plesača označenog brojem* 1 *u smeru kretanja kazaljke na satu). Preostali plesači obrazuju manji krug iz kog opet izlazi k-ti plesač (odbrojava se pocev od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu). Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi* `n`, `k` *($k < n$) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga.*

PRIMER: za $n = 5$, $k = 3$ *redosled izlaska je* 3 1 5 2 4.

9

Programski jezik C

[1]

9.1 Stek

Primer 66 *Provera uparenosti HTML etiketa - stek se implementira preko liste.*

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

/* Maksimalna duzina etikete */
#define MAX_TAG 100

#define OPEN   1
#define CLOSED 2
#define ERROR 0

/* Funkcija ucitava sledecu etiketu i smesta njen naziv u niz s duzine max.
   Vraca OPEN za otvorenu etiketu, CLOSED za zatvorenu etiketu,
   odnosno ERROR inace */
int gettag(char s[], int max)
{   int c, i;
    int type=OPEN;

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

    /* Proveravamo da li je etiketa zatvorena */
    if ((c=getchar())=='/')
       type = CLOSED;
```

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

```
    else
       ungetc(c,stdin);

    /* Citamo etiketu dok nailaze slova i smestamo ih u nisku*/
    for (i=0; isalpha(c=getchar()) && i<max-1; s[i++] = c)
                    ;
    s[i]='\0';

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

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

/*********************************************************************/
/* Stek ce biti implementiran koriscenjem liste */
typedef struct node
{
    char string[MAX_TAG];
    struct node* next;
} NODE;

/* Funkcija postavlja dati string na stek */
void push(NODE** pstack, char* s)
{
    NODE* tmp = (NODE*)malloc(sizeof(NODE));
    if (tmp == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije\n");
        exit(1);
    }
    strcpy(tmp->string, s);
    tmp->next = *pstack;
    *pstack = tmp;
}

/* Funkcija cita podatak sa vrha steka */
char* peek(NODE* stack)
{
    /* Stek ne sme da bude prazan */
    assert(stack != NULL);

    return stack->string;
}

/* Funkcija uklanja podatak sa vrha steka */
void pop(NODE** pstack)
{
    /* Ukoliko je stek prazan ne radimo nista */
    if (*pstack == NULL)
```

```
        return;

    NODE* tmp = (*pstack)->next;
    free(*pstack);
    *pstack = tmp;
}

/* Funkcija proverava da li je dati stek prazan */
int empty(NODE* stack)
{
    return stack == NULL;
}
/* ********************************************************************** */


main()
{
    char tag[MAX_TAG];
    NODE* stack = NULL;

    int type;
    while ((type = gettag(tag,MAX_TAG)) != ERROR)
    {
        if (type == OPEN)
        {
            /* Svaku otvorenu etiketu stavljamo na stek */
            push(&stack, tag);
            printf("Postavio <%s> na stek\n", peek(stack));
        }
        else
        {
            /* Za zatvorene etikete proveravamo da li je stek prazan
            odnosno da li se na vrhu steka nalazi odgovarajuca otvorena etiketa */
            if (!empty(stack) && strcmp(peek(stack), tag) == 0)
            {
                printf("Skidam <%s> sa steka\n", peek(stack));
                /* Uklanjamo etiketu sa steka */
                pop(&stack);
            }
            else
            {
                /* Prijavljujemo gresku */
                printf("Neodgovarajuce : </%s>\n",tag);
                exit(1);
            }
        }
    }

    /* Proveravamo da li je stack ispraznjen */
    if (!empty(stack))
        fprintf(stderr, "Nisu sve etikete zatvorene\n");
}
```

9.2 Drveta

9.2.1 Binarno pretraživačko drvo

Primer 67 *Binarno pretraživačko drvo - drvo sadrži cele brojeve. Ubacivanje realizovano preko* ***

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

/* Struktura jednog cvora drveta */
typedef struct _cvor
{
    /* Podatak */
    int broj;
    /* Pokazivac na levo i desno podstablo */
    struct _cvor *l, *d;
} cvor;

/* Pomocna funkcija koja kreira novi cvor na osnovu datog broja */
cvor* napravi_cvor(int b)
{
    cvor* novi = (cvor*)malloc(sizeof(cvor));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije");
        exit(1);
    }

    /* Postavljamo brojnu vrednost */
    novi->broj = b;

    /* Novi cvor se kreira kao list */
    novi->l = NULL;
    novi->d = NULL;
    return novi;
}

/* Rekurzivna funkcija koja ubacuje dati broj u dato drvo */
void ubaci_u_drvo(cvor** pdrvo, int b)
{
    /* Ukoliko je drvo prazno kreira se novi cvor */
    if (*pdrvo == NULL)
    {
        *pdrvo = napravi_cvor(b);
        return;
    }

    /* Ukoliko je broj koji se ubacuje manji od broja u korenu,
       rekurzivno ga ubacujemo u levo podstablo.
       Ukoliko je broj koji se ubacuje veci od broja u korenu,
       rekurzivno ga ubacujemo u desno podstablo.
    */
```

```
    if (b < (*pdrvo)->broj)
        ubaci_u_drvo(&((*pdrvo)->l), b);
    else if (b > (*pdrvo)->broj)
        ubaci_u_drvo(&((*pdrvo)->d), b);
}


/* Rekurzivna funkcija koja proverava da li dati broj postoji u drvetu */
int pronadji(cvor* drvo, int b)
{
    /* U praznom drvetu ne postoji broj */
    if (drvo == NULL)
        return 0;

    /* Ukoliko je jednak vrednosti u korenu, onda postoji */
    if (drvo->broj == b)
        return 1;

    /* Ukoliko je broj koji trazimo manji od vrednosti u korenu,
       trazimo ga samo u levom podstablu, a inace ga trazimo
       samo u desnom podstablu */
    if (b < drvo->broj)
        return pronadji(drvo->l, b);
    else
        return pronadji(drvo->d, b);
}

/* Rekurzivna funkcija koja ispisuje drvo u inorder redosledu */
void ispisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        ispisi_drvo(drvo->l);
        printf("%d ", drvo->broj);
        ispisi_drvo(drvo->d);
    }
}


/* Rekurzivna funkcija koja uklanja drvo. Obilazak mora biti postorder. */
void obrisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        obrisi_drvo(drvo->l);
        obrisi_drvo(drvo->d);
        free(drvo);
    }
}

main()
```

```
{
    cvor* drvo = NULL;
    ubaci_u_drvo(&drvo, 1);
    ubaci_u_drvo(&drvo, 8);
    ubaci_u_drvo(&drvo, 3);
    ubaci_u_drvo(&drvo, 5);
    ubaci_u_drvo(&drvo, 7);
    ubaci_u_drvo(&drvo, 6);
    ubaci_u_drvo(&drvo, 9);

    if (pronadji(drvo, 3))
        printf("Pronadjeno 3\n");
    if (pronadji(drvo, 2))
        printf("Pronadjeno 2\n");
    if (pronadji(drvo, 7))
        printf("Pronadjeno 7\n");

    ispisi_drvo(drvo);
    putchar('\n');

    obrisi_drvo(drvo);
}
```

Primer 68 *Binarno pretraživacko drvo - drvo sadrži cele brojeve. Ubacivanje realizovano preko eksplicitnog vraćanja korena rezultujućeg drveta.*

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

typedef struct _cvor
{
    int broj;
    struct _cvor *l, *d;
} cvor;

cvor* napravi_cvor(int b)
{
    cvor* novi = (cvor*)malloc(sizeof(cvor));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije");
        exit(1);
    }
    novi->broj = b;
    novi->l = NULL;
    novi->d = NULL;
    return novi;
}

cvor* ubaci_u_drvo(cvor* drvo, int b)
{
    if (drvo == NULL)
        return napravi_cvor(b);
```

```
    if (b < drvo->broj)
        drvo->l = ubaci_u_drvo(drvo->l, b);
    else
        drvo->d = ubaci_u_drvo(drvo->d, b);

    return drvo;
}

/* Funkcija proverava da li dati broj postoji u drvetu */
int pronadji(cvor* drvo, int b)
{
    if (drvo == NULL)
        return 0;

    if (drvo->broj == b)
        return 1;

    if (b < drvo->broj)
        return pronadji(drvo->l, b);
    else
        return pronadji(drvo->d, b);
}

void ispisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        ispisi_drvo(drvo->l);
        printf("%d ", drvo->broj);
        ispisi_drvo(drvo->d);
    }
}


void obrisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        obrisi_drvo(drvo->l);
        obrisi_drvo(drvo->d);
        free(drvo);
    }
}

main()
{
    cvor* drvo = NULL;
    drvo = ubaci_u_drvo(drvo, 1);
    drvo = ubaci_u_drvo(drvo, 8);
    drvo = ubaci_u_drvo(drvo, 5);
    drvo = ubaci_u_drvo(drvo, 3);
```

```
    drvo = ubaci_u_drvo(drvo, 7);
    drvo = ubaci_u_drvo(drvo, 6);
    drvo = ubaci_u_drvo(drvo, 9);

    if (pronadji(drvo, 3))
        printf("Pronadjeno 3\n");
    if (pronadji(drvo, 2))
        printf("Pronadjeno 2\n");
    if (pronadji(drvo, 7))
        printf("Pronadjeno 7\n");

    ispisi_drvo(drvo);
    putchar('\n');

    obrisi_drvo(drvo);
}
```

Primer 69 *Rekurzivne funkcije za rad sa celobrojnim stablima (ne obavezno pretrazivackim): broj_cvorova, broj_listova, suma_cvorova, dubina, najveci_cvor, ...*

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

typedef struct _cvor
{
    int broj;
    struct _cvor *l, *d;
} cvor;

cvor* napravi_cvor(int b)
{
    cvor* novi = (cvor*)malloc(sizeof(cvor));
    if (novi == NULL)
    {
        fprintf(stderr, "Greska prilikom alokacije memorije");
        exit(1);
    }
    novi->broj = b;
    novi->l = NULL;
    novi->d = NULL;
    return novi;
}

void ubaci_u_drvo(cvor** drvo, int b)
{
    if (*drvo == NULL)
    {
        *drvo = napravi_cvor(b);
        return;
    }

    if (b < (*drvo)->broj)
        ubaci_u_drvo(&((*drvo)->l), b);
```

```
    else
        ubaci_u_drvo(&((*drvo)->d), b);
}

void ispisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        ispisi_drvo(drvo->l);
        printf("%d ", drvo->broj);
        ispisi_drvo(drvo->d);
    }
}


void obrisi_drvo(cvor* drvo)
{
    if (drvo != NULL)
    {
        obrisi_drvo(drvo->l);
        obrisi_drvo(drvo->d);
        free(drvo);
    }
}

/* Izracunava sumu svih elemenata u cvorovima drveta */
int suma_cvorova(cvor* drvo)
{
    if (drvo == NULL)
        return 0;
    return  suma_cvorova(drvo->l) +
        drvo->broj +
        suma_cvorova(drvo->d);
}

/* Izracunava broj cvorova datog drveta */
int broj_cvorova(cvor* drvo)
{
    if (drvo == NULL)
        return 0;
    return  broj_cvorova(drvo->l) +
        1 +
        broj_cvorova(drvo->d);
}

/* Izracunava broj listova datog drveta */
int broj_listova(cvor* drvo)
{
    if (drvo == NULL)
        return 0;

    /* Cvor je list ukoliko nema ni jednog naslednika */
```

```
    if (drvo->l == NULL && drvo->d == NULL)
        return 1;

    return  broj_listova(drvo->l) +
        broj_listova(drvo->d);
}

/* Izracunava sumu svih elemenata u listovima drveta */
int suma_listova(cvor* drvo)
{
    if (drvo == NULL)
        return 0;

    if (drvo->l == NULL && drvo->d == NULL)
        return drvo->broj;

    return  suma_listova(drvo->l) +
        suma_listova(drvo->d);
}

/* Ispisuje sve elemente u listovima drveta */
void ispisi_listove(cvor* drvo)
{
    if (drvo == NULL)
        return;

    ispisi_listove(drvo->l);

    if (drvo->l == NULL && drvo->d == NULL)
        printf("%d ", drvo->broj);

    ispisi_listove(drvo->d);
}

/* Izracunava vrednost najveceg cvora u proizvoljnom drvetu.
   Vraca -1 ukoliko je drvo prazno */
int najveci_cvor(cvor* drvo)
{

    if (drvo == NULL)
        return -1;
    else
    {
        int max_l = najveci_cvor(drvo->l);
        int max_d = najveci_cvor(drvo->d);

        return max_l<max_d ?
            (max_d<drvo->broj?drvo->broj:max_d) :
            (max_l<drvo->broj?drvo->broj:max_l);
    }
}
```

```
/* Izracunava dubinu (broj nivoa drveta) */
int dubina(cvor* drvo)
{
    if (drvo == NULL)
        return 0;
    else
    {
        int dl = dubina(drvo->l);
        int dd = dubina(drvo->d);

        return dl<dd ? dd + 1 : dl + 1;
    }
}

main()
{
    cvor* drvo = NULL;
    ubaci_u_drvo(&drvo, 1);
    ubaci_u_drvo(&drvo, 8);
    ubaci_u_drvo(&drvo, 5);
    ubaci_u_drvo(&drvo, 3);
    ubaci_u_drvo(&drvo, 7);
    ubaci_u_drvo(&drvo, 6);
    ubaci_u_drvo(&drvo, 9);

    printf("Suma cvorova : %d\n", suma_cvorova(drvo));
    printf("Broj cvorova : %d\n", broj_cvorova(drvo));
    printf("Broj listova : %d\n", broj_listova(drvo));
    printf("Suma listova : %d\n", suma_listova(drvo));
    printf("Najveci cvor : %d\n", najveci_cvor(drvo));
    printf("Dubina : %d\n", dubina(drvo));
    printf("Listovi : ");
    ispisi_listove(drvo);
    putchar('\n');

    obrisi_drvo(drvo);
}
```

Primer 70 *Program sa ulaza cita tekst i ispisuje broj pojavljivanja svake od reci koje su se javljale u tekstu. Verzija sa binarnim pretrazivackim drvetom.*

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

/* Cvor drveta sadrzi ime reci i broj njenih pojavljivanja */
typedef struct _cvor
{    char ime[80];
     int br_pojavljivanja;
     struct _cvor* levo, *desno;
} cvor;


/* Funkcija ispisuje drvo u inorder redosledu */
```

```
void ispisi_drvo(cvor* drvo)
{    if (drvo!=NULL)
     {  ispisi_drvo(drvo->levo);
        printf("%s %d\n",drvo->ime,drvo->br_pojavljivanja);
        ispisi_drvo(drvo->desno);
     }
}


/* Funkcija uklanja binarno drvo iz memorije */
void obrisi_drvo(cvor* drvo)
{    if (drvo!=NULL)
     {  obrisi_drvo(drvo->levo);
        obrisi_drvo(drvo->desno);
        free(drvo);
     }
}

/* Funkcija ubacuje datu rec u dato drvo i vraca pokazivac na koren drveta */
cvor* ubaci(cvor* drvo, char rec[])
{
    /* Ukoliko je drvo prazno gradimo novi cvor */
    if (drvo==NULL)
    {   cvor* novi_cvor=(cvor*)malloc(sizeof(cvor));
        if (novi_cvor==NULL)
        {  printf("Greska prilikom alokacije memorije\n");
           exit(1);
        }
        strcpy(novi_cvor->ime, rec);
        novi_cvor->br_pojavljivanja=1;
        return novi_cvor;
    }
    int cmp = strcmp(rec, drvo->ime);

    /* Ukoliko rec vec postoji u drvetu uvecavamo njen broj pojavljivanja */
    if (cmp==0)
    {   drvo->br_pojavljivanja++;
        return drvo;
    }

    /* Ukoliko je rec koju ubacujemo leksikografski ispred reci koja je u
       korenu drveta, rec ubacujemo u levo podstablo */
    if (cmp<0)
    {   drvo->levo=ubaci(drvo->levo, rec);
        return drvo;
    }

    /* Ukoliko je rec koju ubacujemo leksikografski iza reci koja je u
       korenu drveta, rec ubacujemo u desno podstablo */
    if (cmp>0)
    {   drvo->desno=ubaci(drvo->desno, rec);
        return drvo;
```

```
    }
}

/* Pomocna funkcija koja cita rec sa standardnog ulaza i vraca njenu
   duzinu, odnosno -1 ukoliko se naidje na EOF */
int getword(char word[], int lim)
{   int c, i=0;
    while (!isalpha(c=getchar()) && c!=EOF)
             ;

    if (c==EOF)
          return -1;
    do
    {   word[i++]=c;
    }while (i<lim-1 && isalpha(c=getchar()));

    word[i]='\0';
    return i;
}

main()
{
   /* Drvo je na pocetku prazno */
   cvor* drvo=NULL;
   char procitana_rec[80];

   /* Citamo rec po rec dok ne naidjemo na kraj datoteke i
      ubacujemo ih u drvo */
   while(getword(procitana_rec,80)!=-1)
      drvo=ubaci(drvo,procitana_rec);

   /* Ispisujemo drvo */
   ispisi_drvo(drvo);

   /* Uklanjamo ga iz memorije */
   obrisi_drvo(drvo);

}
```

Primer 71 *Program koji broji pojavljivanja svih etiketa u HTML datoteci - etikete se ispisuju opadajuci po broju pojavljivanja*

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

/* Struktura cvora drveta u kome se cuvaju etikete zajedno sa brojem
   pojavljivanja. Drvo je pretrazivacko i sortirano je lekikografski po
   etiketama */
typedef struct _node{
    char tag[30];
```

```
    int num;
    struct _node *l,*r;
} node;


/* Zbog sortiranja po broju cvorova, paralelno sa strukturom drveta,
   odrzavamo niz pokazivaca na njegove cvorove */
node* nodes[100];
/* Dosadasnji broj cvorova drveta (razlicitih etiketa) */
int num_nodes = 0;


/* Funkcija kreira cvor koji sadrzi datu etiketu */
node* make_node(char *tag)
{
    node* new_node = (node*) malloc(sizeof(node));
    if(new_node == NULL)
    {
        fprintf(stderr,"Greska prilikom alokacije memorije\n");
        exit(1);
    }

    strcpy(new_node->tag, tag);
    new_node->l=NULL;
    new_node->r=NULL;
    new_node->num = 1;

    /* Dopisujemo cvor u niz postojecih cvorova */
    nodes[num_nodes++] = new_node;

    return new_node;
}

/* Funkcija umece datu etiketu u postojece drvo. Ukoliko etiketa postoji,
   povecava se njen broj pojavljivanja */
void insert(node** ptree, char tag[])
{
    int cmp;
    if(*ptree==NULL)
    {
        *ptree = make_node(tag);
        return;
    }

    cmp = strcmp(tag, (*ptree)->tag);

    if (cmp < 0)
        insert(&((*ptree)->l), tag);
    else if (cmp > 0)
        insert(&((*ptree)->r), tag);
    else
        (*ptree)->num++;
```

```
}

/* Funcija za ispis drveta */
void print(node* tree)
{
    if(tree != NULL)
    {
        print(tree->l);
        printf("%-10s - %3d\n", tree->tag, tree->num);
        print(tree->r);
    }
}

/* Funkcija koja uklanja drvo */
void remove_tree(node* tree)
{
    if(tree != NULL)
    {
        remove_tree(tree->l);
        remove_tree(tree->r);
        free(tree);
    }
}


/* Funkcija poredjenja za poziv ugradjene funkcije qsort. Porede se
   dva cvora drveta na osnovu broja pojavljivanja etiketa */
int compare(const void* pa, const void* pb)
{
    return (*((node**)pb))->num - (*((node**)pa))->num;
}

/* Funkcija ucitava etiketu iz date datoteke. Funkcija vraca
   logicku vrednost koja indikuje da li je etiketa uspesno
   procitana */
int get_tag(FILE* f, char tag[])
{
    int c;
    int i = 0;

    /* Preskacemo sve do prvog znaka '<' ili kraja */
    while ((c = fgetc(f)) != EOF && c != '<')
        ;

    /* Nije bilo vise etiketa */
    if (c == EOF)
        return 0;

    /* Citamo prvi karakter etiketa*/
    c = fgetc(f);

    /* Gutamo / kod zatvorenih etiketa */
```

```
    if (c == '/')
        c = fgetc(f);

    /* Ime etikete cine slova */
    while(isalpha(c))
    {
        tag[i++] = c;
        c = fgetc(f);
    }
    tag[i] = '\0';

    /* Preskacemo sve do > ili do kraja */
    while (c != '>' && c != EOF)
        c = fgetc(f);

    if (c == EOF)
        return 0;

    return 1;
}

main(int argc, char* argv[])
{
    /* Tekuca procitana etiketa */
    char tag[30];

    /* Drvo koje je leksikografski sortirano zbog brze pretrage */
    node* tree = NULL;

    /* Datoteka iz koje se cita */
    FILE* f;
    int i;

    /* Proverava se korektnost argumenata komandne linije */
    if (argc < 2)
    {
        printf("Upotreba : %s ime_datoteke\n", argv[0]);
        exit(1);
    }

    /* Otvara se datoteka */
    f = fopen(argv[1], "r");
    if (f == NULL)
    {
        fprintf(stderr, "Greska prilikom otvaranja %s\n", argv[1]);
        return;
    }

    /* Kreiramo drvo na osnovu sadrzaja datoteke */
    while(get_tag(f, tag) == 1)
        insert(&tree, tag);
```

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

    /* Sortiramo cvorove niza na osnovu broja pojavljivanja etiketa */
    qsort(nodes, num_nodes, sizeof(node*), &compare);

    /* Ispisujemo sortirane cvorove */
    for (i = 0; i<num_nodes; i++)
        printf("%-10s - %3d\n", nodes[i]->tag, nodes[i]->num);

    /* Uklanjamo drvo */
    remove_tree(tree);

}
```

9.3 Zadaci za vežbu

Zadatak 32 Septembar, 2005. *Napisati program koji na standardni izlaz ispisuje naziv (BEZ ATRIBUTA) najčešće korišćene etikete u datoteci ulaz.htm. Ako ima više takvih, ispisati ma koju. Koristiti uređeno binarno stablo. Pretpostaviti da je ulazna datoteka sintaksno korektna.*

Zadatak 33 Drugi kolokvijum za II tok 2004.godine - rad na računaru *Napisati program koji iz tekstualne datoteke čiji je put dat u argumentu komandne linije učitava različite prirodne brojeve i:*

1. *dodaje ih redom u uredjeno binarno stablo*
2. *u dobijenom drvetu izračunava dužinu najdužeg puta od korena do nekog lista i*
3. *štampa u rastućem poretku (bez ponavljanja) sve brojeve koji su nalaze na putevima te dužine od korena do listova.*

10

Programski jezik C

[1]

10.1 Grafovi

Graf `G=(V,E)` *sastoji se od skupa* `V` *čvorova i skupa* `E` *grana. Grane predstavljaju relacije između čvorova i odgovara paru čvorova. Graf može biti usmeren (orijentisan), ako su mu grane uređeni parovi i neusmeren (neorjentisan) ako su grane neuređeni parovi.*

Uobičajena su dva načina predstavljanja grafova. To su matrica povezanosti *grafa i* lista povezanosti.

Matrica povezanosti je kvadratna matrica dimenzije `n`*, pri čemu je* `n` *broj čvorova u grafu, takva da je element na preseku* `i`*-te vrste i* `j`*-te kolone jednak jedinici ukoliko postoji grana u grafu od* `i`*-tog do* `j`*-tog čvora, inače je nula.*

Umesto da se i sve nepostojeće grane eksplicitno predstavljaju u matrici povezanosti, mogu se formirati povezane liste od jedinica iz `i`*-te vrste za* `i=1,2,...,n`*. To je lista povezanosti. Svakom čvoru se pridružuje povezana lista, koja sadrži sve grane susedne tom čvoru. Graf je predstavljen vektorom lista. Svaki elemenat vektora sadrži ime (indeks) čvora i pokazivač na njegovu listu čvorova.*

Prvi problem na koji se nailazi pri konstrukciji bilo kog algoritma za obradu grafa je kako pregledati ulaz. Postoje dva osnovna algoritma za obilazak grafa: pretraga u dubinu (`DFS`*, skraćenica od depth-first-search) i pretraga u širinu (*`BFS`*, skraćenica od breadth-first-search).*

Kod `DFS` *algoritma, obilazak započinje iz proizvoljnog zadatog čvora* `r` *koji se naziva* koren pretrage u dubinu*. Koren se označava kao posećen. Zatim se bira proizvoljan neoznačen čvor* `r1`*, sussedan sa* `r`*, pa se iz čvora* `r1` *rekurzivno startuje pretraga u dubinu. Iz nekog nivoa rekurzije izlazi se kad se naiđe na čvor v kome su svi susedi već označeni.*

Primer 72 *Primer reprezentovanja grafa preko matrice povezanosti. U programu se unosi neorijentisan graf i DFS algoritmom se utvrdjuju čvrovi koji su dostižni iz cvora 0.*

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

int** alociraj_matricu(int n)
{   int **matrica;
    int i;
    matrica=malloc(n*sizeof(int*));
```

[1] Zasnovano na materijalu `Algoritmi, Miodrag Živković` i `http://www.matf.bg.ac.yu/~filip`

```
    for (i=0; i<n; i++)
       matrica[i]=calloc(n,sizeof(int));
    return matrica;
}

void oslobodi_matricu(int** matrica, int n)
{   int i;
    for (i=0; i<n; i++)
        free(matrica[i]);
    free(matrica);
}

int* alociraj_niz(int n)
{   int* niz;
    niz=calloc(n,sizeof(int));
    return niz;
}

void oslobodi_niz(int* niz)
{   free(niz);
}

void unesi_graf(int** graf, int n)
{     int i,j;
      for (i=0; i<n; i++)
         for (j=i; j<n; j++)
         {   printf("Da li su element %d i %d povezani : ",i,j);
             do
             {  scanf("%d",&graf[i][j]);
                graf[j][i]=graf[i][j];
             } while (graf[i][j]!=0 && graf[i][j]!=1);
         }
}

void ispisi_graf(int** graf, int n)
{     int i,j;
      for (i=0; i<n; i++)
      {  for (j=0; j<n; j++)
             printf("%d",graf[i][j]);
         printf("\n");
      }
}



/* Broj cvorova grafa (dimenzija matrice) */
int n;
/* Matrica povezanosti */
int **graf;


/* Pomocni vektor koji govori o tome koji su cvorovi posecivani
```

```
   tokom DFS obilaska */
int *posecen;

/* Rekurzivna implementacija DFS algoritma */
void poseti(int i)
{    int j;
     posecen[i]=1;
     printf("Posecujem cvor %d\n",i);
     for (j=0; j<n; j++)
        if (graf[i][j] && !posecen[j])
               poseti(j);
}


main()
{    int i, j;
     printf("Unesi broj cvorova : ");
     scanf("%d",&n);

     graf=alociraj_matricu(n);
     unesi_graf(graf,n);
     ispisi_graf(graf,n);

     posecen=alociraj_niz(n);
     poseti(0);

     oslobodi_niz(posecen);
     oslobodi_matricu(graf,n);

}
```

Primer 73 *Primer predstavljanja grafa preko niza listi suseda svakog od čvorova grafa U programu se unosi graf i DFS algoritmom se utvrdjuje koji su čvorovi dostižni iz cvora 0.*

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

/* Cvor liste suseda */
typedef struct _cvor_liste
{  int broj; /* Indeks suseda */
   struct _cvor_liste* sledeci;
} cvor_liste;

/* Ubacivanje na pocetak liste */
cvor_liste* ubaci_u_listu(cvor_liste* lista, int broj)
{   cvor_liste* novi=malloc(sizeof(cvor_liste));
    novi->broj=broj;
    novi->sledeci=lista;
    return novi;
}


/* Brisanje liste */
```

```
void obrisi_listu(cvor_liste* lista)
{   if (lista)
    {  obrisi_listu(lista->sledeci);
       free(lista);
    }
}

/* Ispis liste */
void ispisi_listu(cvor_liste* lista)
{   if (lista)
    {  printf("%d ",lista->broj);
       ispisi_listu(lista->sledeci);
    }
}

/* Graf predstavlja niz pokazivaca na pocetke listi suseda */
#define MAX_BROJ_CVOROVA 100
cvor_liste* graf[MAX_BROJ_CVOROVA];
int broj_cvorova;


/* Rekurzivna implementacija DFS algoritma */
int posecen[MAX_BROJ_CVOROVA];
void poseti(int i)
{   cvor_liste* sused;
    printf("Posecujem cvor %d\n",i);
    posecen[i]=1;
    for( sused=graf[i]; sused!=NULL; sused=sused->sledeci)
        if (!posecen[sused->broj])
              poseti(sused->broj);
}


main()
{   int i;
    printf("Unesi broj cvorova grafa : ");
    scanf("%d",&broj_cvorova);
    for (i=0; i<broj_cvorova; i++)
    {    int br_suseda,j;

         graf[i]=NULL;

         printf("Koliko cvor %d ima suseda : ",i);
         scanf("%d",&br_suseda);
         for (j=0; j<br_suseda; j++)
         {    int sused;
              do
              {
                printf("Unesi broj %d.-tog suseda cvora %d : ",j,i);
                scanf("%d",&sused);
              } while (sused<1 && sused>broj_cvorova);
              graf[i]=ubaci_u_listu(graf[i],sused-1);
```

```
        }
    }

    for (i=0; i<broj_cvorova; i++)
    {   printf("%d - ",i);
        ispisi_listu(graf[i]);
        printf("\n");
    }

    poseti(0);
}
```

Primer 74 *MINESWEEPER - primer jednostavne igrice. Program demonstrira rad sa matricama, slučajnim brojevima i rekurzivnu implementaciju DFS algoritma za obilazak grafova.*

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

/* Dimenzija table */
int   n;

/* Tabla koja sadrzi 0 i 1 u zavisnosti od toga da li na polju postoji bomba */
int** bombe;

/* Tabla koja opisuje tekuce stanje igre. Moze da sadrzi sledece vrednosti :
    ZATVORENO - opisuje polje koje jos nije bilo otvarano
    PRAZNO - polje na kome ne postoji ni jedna bomba
    broj od 1-8 - polje koje je otvoreno i na kome pise koliko bombi postoji u okolini
    ZASTAVICA - polje koje je korisnik oznacio zastavicom
*/
#define PRAZNO (-1)
#define ZATVORENO 0
#define ZASTAVICA 9

int** stanje;

/* Ukupan broj bombi */
int broj_bombi;

/* Ukupan broj postavljenih zastavica */
int broj_zastavica = 0;


/* Pomocne funkcije za rad sa matricama */
int** alociraj(int n)
{  int i;
   int** m=malloc(n*sizeof(int*));
   for (i=0; i<n; i++)
       m[i]=calloc(n,sizeof(int));
   return m;
}
```

```
void obrisi(int** m, int n)
{   int i;
    for (i=0; i<n; i++)
       free(m[i]);

    free(m);
}

/* Funkcija postavlja bombe */
void postavi_bombe()
{  broj_bombi=(n*n)/6;
   int kolona;
   int vrsta;
   int i;

   /* Inicijalizujemo generator slucajnih brojeva */
   srand(time(NULL));

   for (i=0; i<broj_bombi; i++)
   {    /* Racunamo slucajni polozaj bombe */
        kolona=rand()%n;
        vrsta=rand()%n;


        /* Ukoliko bomba vec postoji tu, opet idemo u istu iteraciju */
        if (bombe[vrsta][kolona]==1)
        {   i--;
            continue;
        }

        /* Postavljamo bombu */
        bombe[vrsta][kolona]=1;
  }
}


/* Funkcija ispisuje tablu sa bombama */
void ispisi_bombe()
{   int i,j;
    for (i=0; i<n; i++)
    {   for (j=0; j<n; j++)
            printf("%d",bombe[i][j]);
        printf("\n");
    }
}

/* Funkcija ispisuje tekuce stanje */
void ispisi_stanje()
{   int i,j;

    /* Brisemo ekran pozivajuci komandu operativnog sistema */
```

```
    system("clear");


    for (i=0; i<n; i++)
    {   for (j=0; j<n; j++)
        {    if (stanje[i][j]==ZATVORENO)
                  printf(".");
             else if (stanje[i][j]==PRAZNO)
                  printf(" ");
             else if (stanje[i][j]==ZASTAVICA)
                  printf("*");
             else
                  printf("%d",stanje[i][j]);
        }
        printf("\n");
    }

}

/* Funkcija postavlja zastavicu na dato polje ili je uklanja
   ukoliko vec postoji */
void postavi_zastavicu(int i, int j)
{    if (stanje[i][j]==ZATVORENO)
     {    stanje[i][j]=ZASTAVICA;
          broj_zastavica++;
     }
     else if (stanje[i][j]==ZASTAVICA)
     {    stanje[i][j]=ZATVORENO;
          broj_zastavica--;
     }
}

/* Funkcija izracunava koliko bombi postoji u okolini date bombe */
int broj_bombi_u_okolini(int v, int k)
{    int i, j;
     int br=0;
     /* Prolazimo kroz sva okolna polja */
     for (i=-1; i<=1; i++)
         for(j=-1; j<=1; j++)
         {  /* preskacemo centralno polje */
            if (i==0 && j==0)
                continue;
            /* preskacemo polja "van table" */
            if (v+i<0 || k+j<0 || v+i>=n || k+j>=n)
                continue;
            if (bombe[v+i][k+j]==1)
                br++;
         }

   return br;

}
```

```
/* Centralna funkcija koja vrsi otvaranje polja i pritom se otvaranje "siri"
   i na polja koja su oko datog */

void otvori_polje(int v, int k)
{   /* Ukoliko smo "nagazili" bombu zavrsavamo program */
    if (bombe[v][k]==1)
    {   printf("BOOOOOOOOOOOOOOOOM!!!!\n");
        ispisi_bombe();
        exit(1);
    }
    else
    {    /* Brojimo bombe u okolini */
         int br=broj_bombi_u_okolini(v,k);

         /* Azuriramo stanje ovog polja */
         stanje[v][k]=(br==0)?PRAZNO:br;

         /* Ukoliko u okolini nema bombi, rekurzivno otvaramo
            sva polja u okolini koja su zatvorena */
         if (br==0)
         {   /* Petlje indeksiraju sva okolna polja */
             int i,j;
             for (i=-1; i<=1; i++)
                for (j=-1; j<=1; j++)
                {   /* Preskacemo centralno polje */
                    if (i==0 && j==0)
                        continue;
                    /* Preskacemo polja van table */
                    if (v+i<0 || v+i>=n || k+j<0 || k+j>=n)
                          continue;
                    /* Ukoliko je okolno polje zatvoreno, otvaramo ga */
                    if (stanje[v+i][k+j]==ZATVORENO)
                          otvori_polje(v+i, k+j);
                }
         }
    }
}


/* Funkcija utrdjuje da li je partija gotova
   Partija je gotova u trenutku kada su sve bombe pokrivene zastavicama i
   kada nijedno drugo polje nije pokriveno zastavicom
*/

int gotova_partija()
{  int i,j;
   for (i=0; i<n; i++)
      for (j=0; j<n; j++)
      {   /* Ukoliko postoji nepokrivena bomba, partija nije zavrsena */
          if (bombe[i][j]==1 && stanje[i][j]!=ZASTAVICA)
              return 0;
```

```
      }

  /* Partija je zavrsena samo ukoliko je broj zastavica jednak broj bombi */
  return broj_zastavica==broj_bombi;

}

main()
{

  /* Unosimo dimenziju table */
  printf("Unesite dimenziju table : ");
  scanf("%d",&n);


  /* Alociramo table */
  bombe=alociraj(n);
  stanje=alociraj(n);

  /* Postavljamo bombe */
  postavi_bombe();

  /* Sve dok partija nije gotova */
  while(!gotova_partija())
  {    int v,k;
       char akcija;

       /* Ispisujemo tekuce stanje */
       ispisi_stanje();

       /* Sve dok korisnik ne unese o ili z trazimo od njega da upise odgovarajucu akciju */
       do
       {  printf("Unesi akciju (o - otvaranje polja, z - postavljanje zastavice) : ");
      while(isspace(akcija = getchar()));
       }  while (akcija!='o' && akcija!='z');

       /* Trazimo od korisnika da unese koordinate polja sve dok ih ne unese ispravno
          Korisnicke koordinate krecu od 1, a interne od 0 */
       do
       {
          printf("Unesi koordinate polja : ");
          scanf("%d",&v);
          scanf("%d",&k);
       } while(v<1 || v>n || k<1 || k>n);

       /* Reagujemo na akciju */
       switch(akcija)
       {   case 'o':
               otvori_polje(v-1,k-1);
               break;
           case 'z':
               postavi_zastavicu(v-1,k-1);
```

```
        }
    }

    /* Konstatujemo pobedu */
    ispisi_stanje();
    obrisi(stanje, n);
    obrisi(bombe, n);

    printf ("Cestitam! Pobedili ste\n");

}
```

11

Programski jezik C

11.1 Zadaci sa prethodnih ispita i kolokvijuma iz Osnova Programiranja

Osnovi programiranja, februar 2006. - prva grupa

1. *Ime datoteke zadaje se iz komandne linije. Napisati program koji ispisuje sadržaj datoteke na sledeći način: redni broj prvog znaka u liniji, a zatim osam po osam znakova u redu, i to heksadecimalno i "karakterski" kao u donjem primeru:*

```
 0  23 69 6E 63 6C 75 64 65     #include
 8  20 3C 73 74 64 69 6F 73     <stdio.h
16  68 3E 0D 0A 23 69 6E 63      >  #incl
```

2. *Definišemo strukturu VREME na sledeći način:*

```
typedef struct{
int sat, min, sek;
} VREME;
```

(a) Napisati funkciju sa protipom `VREME *napravi(int sat, int min, int sek )` *koja dinamički alocira memorijski prostor u koji će smestiti strukturu VREME, inicijalizovanu vrednostima koje se prenose kao parametri. Funkcija vraća pokazivač na kreiranu strukturu. (b) Sastaviti funkciju sa prototipom* `void plus(VREME *t)` *koja povećava za jednu sekundu vreme predstavljano strukturom t.*

3. *Napisati program koji za dato $n \leq 15$ ispisuje prvih n redova trougla od Stirlingovih brojeva I vrste $s(n, m)$, $1 \leq m \leq n$. Stirlingovi brojevi I vrste zadaju se rekurentnom relacijom*

$$s(n+1,m) = \begin{cases} -ns(n,m), & m = 1 \\ s(n,m-1) - ns(n,m), & 1 < m \leq n \\ s(n,m-1), & m = n+1 \end{cases}$$

pri čemu je $s(1,1) = 1$. Koristiti jedan jednodimenzionalni niz. Ispis treba da bude sledećeg oblika:

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

```
-6 11 -6 1
24 -50 35 -10 1
.................
```

4. *Napisati funkciju sa jednim argumentom* `n` *tipa* `int` *koja vraća razliku broja jedinica na parnim i neparnim pozicijama u binarnom zapisu argumenta.*

   PRIMER: za $n = 19 = (10011)_2$ *izlaz je* 1.

5. *Grupa od n plesača (na čijim kostimima su u smeru kazaljke na satu redom brojevi od* 1 *do n) izvodi svoju plesnu tačku tako što formiraju krug iz kog najpre izlazi k-ti plesač (odbrojava se počev od plesača označenog brojem* 1 *u smeru kretanja kazaljke na satu). Preostali plesači obrazuju manji krug iz kog opet izlazi k-ti plesač (odbrojava se pocev od sledećeg suseda prethodno izbačenog, opet u smeru kazaljke na satu). Izlasci iz kruga se nastavljaju sve dok svi plesači ne budu isključeni. Celi brojevi* `n`, `k` *(*$k < n$*) se učitavaju sa standardnog ulaza. Napisati program koji će na standardni izlaz ispisati redne brojeve plesača u redosledu napuštanja kruga.*

   PRIMER: za $n = 5$, $k = 3$ *redosled izlaska je* 3 1 5 2 4.

Osnovi programiranja, februar 2006. - druga grupa

1. *Imena dveju datoteka iste veličine zadaju se iz komandne linije. Napisati program koji upoređuje sadržaje datoteka. Ako je i-ti znak u prvoj datoteci* a_i, *a i-ti znak u drugoj datoteci* b_i, *onda program izračunava znakove*

$$c_i = \begin{cases} a_i, & ako a_i = b_i, \text{ a } a_i \text{ nije kontrolni znak – sa ASCII kodom} < 32 \\ \text{' '}, & \text{ako } a_i = b_i, \text{ a } a_i \text{ jeste kontrolni znak'.', ako } a_i \neq b_i \end{cases}$$

   Znakove c_i *program ispisuje na standardni izlaz, po* 16 *znakova u jednom redu, pri čemu svaki red počinje rednim brojem prvog znaka u redu.*

```
  datoteka 1:          datoteka 2:            izlaz:

Imena dveju dato     Imena dve datote       1  Imena dve.....t.
teka iste velici     ke  iste  velici      17  .e....... velici
ne zadaju se iz      ne zadaju se iz        33  ne zadaju se iz
```

2. *Definišemo strukturu VREME na sledeći način:*

```
typedef struct{
    int sat, min, sek;
    } VREME;
```

   (a) Napisati funkciju sa protipom `VREME *napravi(int sat, int min, int sek )` *koja dinamički alocira memorijski prostor u koji će smestiti strukturu VREME, inicijalizovanu vrednostima koje se prenose kao parametri. Funkcija vraća pokazivač na kreiranu strukturu. (b) Sastaviti funkciju sa prototipom* `void plus(VREME *t)` *koja povećava za jednu sekundu vreme predstavljano strukturom t.*

3. *Napisati program koji za dato* $n \leq 15$ *ispisuje prvih n redova trougla od Stirlingovih brojeva II vrste* $S(n, m)$, $1 \leq m \leq n$. *Stirlingovi brojevi II vrste zadaju se rekurentnom relacijom* $S(n, k) = S(n-1, k-1) + kS(n-1, k)$, $1 < k < n$ *pri čemu je* $S(n, 1) = S(n, n) = 1$ *Koristiti jedan jednodimenzionalni niz. Ispis treba da bude sledećeg oblika:*

```
      1
      1      1
      1      3      1
      1      7      6      1
      1     15     25     10      1
      1     31     90     65     15      1
. . . . . . . . . . . . . . .
```

4. *Napisati funkciju sa jednim argumentom* `n` *tipa* `int` *koja vraća razliku broja jedinica na* 16 *viših i* 16 *nižih pozicija (koeficijenti uz* 2^0, 2^1, ..., 2^{15}*) u binarnom zapisu argumenta. Pretpostaviti da je argument veličine* 4 *bajta (*32 *bita).*

   PRIMER: za $n = 7 \times 2^{16} + 3 = (11100000000000011)_2$ *izlaz je* 1.

5. *Na osnovu niza* `a` *dužine* `n`*, koji sadrži neku permutaciju brojeva* $0, 1, \ldots, n-1$*, može se izračunati niz* `b` *iste dužine na sledeći način:*

   - `b[0]` *je indeks broja* `0` *u* `a`*;* `0` *se briše iz* `a`*; dužina* `a` *postaje* $n-1$*;*
   - `b[1]` *je indeks broja* `1` *u* `a`*;* `1` *se briše iz* `a`*; dužina* `a` *postaje* $n-2$*;*
   - `b[2]` *je indeks broja* `2` *u* `a`*;* `2` *se briše iz* `a`*; dužina* `a` *postaje* $n-3$*;*
   - ...

   Napisati funkciju `void tranperm(int n, int a[], int b[])` *koja za dati niz* `a` *(permutaciju) izračunava niz* `b`*. Pri tome treba izbeći pomeranja članova niza* `a`*.*

   PRIMER: za $n = 5$*,* `a={3,5,0,4,2,1}` *rezultat treba da bude* `b={2,4,3,0,1,0}`

Zadatak 34 januar 2006.(I grupa) *Napisati funkciju* ***int triplcmp(const char *s, const char *t)*** *za poređenje, prema dekadnoj vrednosti, dva heksadekadna tripleta* `s` *i* `t` *kojima su predstavljene dve boje* `RGB` *modela (heksadekadni triplet je oblika* `#xxxxxx`*, gde je* `x` *- heksadekadna cifra). Funkcija treba da vrati vrednost* `-1` *ako je* `s < t`*,* `0` *ako je* `s = t` *i* `1` *ako je* `s > t`*. Na primer, za* `s=#FFFFFF`*,* `t=#aa00ee`*, funkcija treba da vrati vrednost* `1`*. (Za triplet* `#aa00ee` *dekadna vrednost je* $10 * 16^5 + 10 * 16^4 + 14 * 16 + 14 = 11.141.358$*).)*

Zadatak 35 januar 2006.(I grupa) *Napisati program koji će iz datoteke* `seminarski.htm` *prepisati nazive međusobno različitih etiketa (bez atributa) u binarno stablo pretrage, a na standardni izlaz ispisati ukupan broj listova drveta. Pretpostaviti da naziv etikete nije duži od* `30` *karaktera.*

Zadatak 36 januar 2006.(I grupa) *Napisati funkciju koja za celobrojni niz dimenzije* `n`*, proverava da li među elementima niza postoje neka dva koja su jednaka.*

Zadatak 37 januar 2006.(I grupa)

1. *Napisati funkciju* ***void propol (int n, double a[], int m, double b[], int *k, double c[])*** *čiji su argumenti* `a` *i* `b` *nizovi koeficijenata polinoma stepena* `n` *i* `m`*, redom. Funkcija izračunava elemente niza* `c` *koeficijenata polinoma koji se dobije množenjem polinoma* `a` *i* `b`*, i stepen* `k` *proizvoda.*

2. *Napisati program koji iz datoteke ulaz.txt učitava dva polinoma (stepen prvog polinoma, pa njegovi koeficijenti, počev od slobodnog člana; stepen drugog polinoma, pa njegovi koeficijenti), izračunava njihov proizvod i na standardni izlaz štampa stepen i koeficijente proizvoda.*

Zadatak 38 januar 2006.(I grupa) *Neka je broj* n_1 *proizvod cifara datog broja* `n`*, broj* n_2 *proizvod cifara broja* n_1*,..., broj* n_k *proizvod cifara broja* n_{k-1}*, pri čemu je* `k` *najmanji prirodan broj za koji je* n_k *jednocifren. Napisati funkciju koja za dato* `n` *izračunava* `k`*. Na primer, vrednosti ove funkcije od* `10, 25, 39` *su redom* `1, 2, 3`*.*

Zadatak 39 januar 2006.(II grupa) *Napisati funkciju koja u datom celobrojnom nizu* `A` *dužine* `n` *pronalazi (ako postoji) takav par indeksa* `(i,j)` *da je zbir članova niza sa indeksima od* `i` *do* `j` *jednak zadatom broju* `m`.

Zadatak 40 januar 2006.(II grupa) *Napisati program koji će iz datoteke čije se ime unosi kao argument komandne linije prepisati nazive međusobno različitih zatvorenih etiketa u binarno stablo pretrage, a na standardni izlaz ispisati dubinu stabla. Pretpostaviti da zatvorena etiketa počinje sa "* $</$ *" i da naziv etikete nije duži od* `30` *karaktera.*

Zadatak 41 januar 2006.(II grupa) *Napisati funkciju koja za dve niske koje se prenose kao parametri utvrđuje da li su anagrami ili ne. Dve niske su anagrami ako se sastoje od istog broja istih karaktera. Na primer, niske "anagram" i "ramgana" jesu anagrami, dok "anagram" i "angrm" nisu.*

Zadatak 42 januar 2006.(II grupa)

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

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

Zadatak 43 januar 2006.(II grupa) *Hemingovo rastojanje dva cela nenegativna broja jednako je broju cifara u binarnom zapisu tih brojeva, koje su na istim pozicijama a razlikuju se. Na primer, Hemingovo rastojanje brojeva* $15 = (1111)_2 = (01111)_2$ *i* $27 = (11011)_2$ *je* `2`. *Napisati funkciju koja izračunava Hemingovo rastojanje dva zadata cela nenegativna broja.*

Zadatak 44 I kolokvijum, 18.januar 2006.(I grupa) *Napisati program pomoću kojeg se za dati broj n izračunava* `n`*-ti član niza* $F_n = 3 * F_{n-1} - 2 * F_{n-2} + F_{n-1} * F_{n-2}$ *pri čemu je* $F_0 = 1$ *i* $F_1 = 1$. *U programu ne koristiti nizove.*

Zadatak 45 I kolokvijum, 18.januar 2006.(I grupa)

1. *Napisati funkciju* ***unsigned izdvoj_n(unsigned x, unsigned n)*** *za izračunavanje broja koji se dobija od* `n` *krajnjih desnih bitova broja* `x`. *Na primer, ako je* $x = 54(00...00110110)$, *a* $n = 3$, *tada funkcija treba da vrati broj* $6(00...00000110)$.*)*

2. *Napisati program koji, za učitane vrednosti* `x`, `n` *poziva funkciju* ***izdvoj_n*** *i na standardni izlaz izdaje rezultat.*

Zadatak 46 I kolokvijum, 18.januar 2006.(I grupa) *Neka je dat niz* `X` *od* `N` *nenegativnih celih brojeva. Sastaviti funkciju koja će iz niza* `X` *izbacivati sva pojavljivanja broja* `0` *i popunjavati ta mesta u nizu tako što će se preostali elementi niza pomerati ka početku niza. Odrediti i novu dimenziju* `N` *niza* `X`. *Npr. ulaz:* `N = 10, X = 0 22 11 2 0 17 33 4 0 999` $\rightarrow$ *izlaz :* `N = 7, X = 22 11 2 17 33 4 999`.

Zadatak 47 I kolokvijum, 18.januar 2006.(I grupa) *Napisati* `C` *program koji kreira i na standardni izlaz izdaje opis* `HTML` *tabele sa tri kolone. Zaglavlja kolona su redom niske:* n, kvadrat, kub. *U prvoj koloni se nalaze vrednosti od* `1..15`, *a u drugoj i trećoj koloni su kvadrati i kubovi tih vrednosti, redom.*

Zadatak 48 I kolokvijum, 18.januar 2006.(I grupa) *Danas je sreda,* `18.januar 2006 godine`. *Napisati funkciju koja za zadati datum (dan, redni broj meseca, godina, posle* `1.1.1900.godine`*) određuje dan u nedelji. Na primer, za trojku* `(19,1,2006)` *funkcija treba da vrati broj* `4`.

Zadatak 49 I kolokvijum, 18.januar 2006.(II grupa) *Napisati program pomoću kojeg se za dati broj n izračunava* `n`*-ti član niza* $F_n = 2 * F_{n-1} * F_{n-2} - 6 * F_{n-1} + F_{n-2}^2$ *pri čemu je* $F_0 = 2$ *i* $F_1 = 3$. *U programu ne koristiti nizove.*

Zadatak 50 I kolokvijum, 18.januar 2006.(II grupa)

1. *Napisati funkciju* ***unsigned izbaci_n(unsigned x, unsigned n)*** *za izračunavanje broja koji se dobija brisanjem* `n` *krajnjih desnih bitova broja* `x`. *Na primer, ako je* $x = 54(00...00110110)$, *a* $n = 3$, *tada funkcija treba da vrati broj* $48(00...00110000)$.*)*

2. *Napisati program koji, za učitane vrednosti* `x`, `n` *poziva funkciju* ***izdvoj_n****i na standardni izlaz izdaje rezultat.*

Zadatak 51 I kolokvijum, 18.januar 2006.(I grupa) *Neka je dat niz* `X` *od* `N` *nenegativnih celih brojeva. Sastaviti funkciju koja će iz niza* `X` *izbacivati sva pojavljivanja negativnih brojeva i popunjavati ta mesta u nizu tako što će se preostali elementi niza pomerati ka početku niza. Odrediti i novu dimenziju* `N` *niza* `X`. *Npr. ulaz:* `N = 6, X = 0 -2 11 0 -333` → *izlaz :* `N = 4, X = 0 11 0 0`.

Zadatak 52 I kolokvijum, 18.januar 2006.(II grupa)

1. *Napisati funkciju* ***int palindrom(int broj)*** *koja proverava da li je broj palindrom i vraća vrednost* `1` *ako jeste,* `0` *ako nije. Na primer, brojevi* `1, 44, 121, 112211, 12321, i 5665` *jesu palindromi, a brojevi* `123, 67, 8908` *nisu.*

2. *Napisati program koji proverava da li je uneti broj palindrom.*

Zadatak 53 I kolokvijum, 18.januar 2006.(II grupa) *Napisati funkciju koja na standardni izlaz ispisuje sve linkove iz* `HTML` *dokumenta sadržanog u datoj nisci* `s`. *Na primer, u delu niske* `s`

```
<A HREF="http://www.bg.ac.yu">
<IMG SRC="/images/univplavi.gif" ALT="Univerzitet u Beogradu" BORDER=0></A>
```

funkcija treba da pronađe

`http://www.bg.ac.yu`.

Zadatak 54 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 55 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 56 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 57 Septembar, 2005. *Napisati program koji na standardni izlaz ispisuje naziv (BEZ ATRIBUTA) najčešće korišćene etikete u datoteci ulaz.htm. Ako ima više takvih, ispisati ma koju. Koristiti uređeno binarno stablo. Pretpostaviti da je ulazna datoteka sintaksno korektna.*

Zadatak 58 Septembar, 2005.

1. *Napisati funkciju* `int poredi(char* p, char* d)` *koja vraća* `-1` *ukoliko je* `p<d`, `0` *ukoliko je* `p == d` *a 1 ako je* `p>d`, *pri čemu su* `p` *i* `d` *dva velika cela neoznačena broja zadata nizom(niskom) svojih cifara.*

2. *Argumenti komandne linije su imena dve datoteke koje sadrže cele neoznačene brojeve (po jedan u svakoj liniji, sa maksimalno 1000 cifara) sortirane u rastućem poretku po numeričkoj vrednosti. Broj linija nije unapred poznat.*

Napisati program koji upisuje sadržaj ove dve datoteke u datoteku `Spoj.txt` *tako da i ona bude sortirana.*

Zadatak 59 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 60 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 61 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 62 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 63 Jun, 2004. *Napisati funkciju koja koja kao argumente prihvata dve niske i proverava da li se prva od zadatih niski može dobiti cikličnim pomeranjem karaktera druge niske.*

Zadatak 64 Jun, 2004. Igrupa *Data je datotka* `brojevi.txt` *koja sadrži cele brojeve, po jedan u svakom redu.*

1. *Napisati funkciju koja iz zadate datoteke učitava brojeve i smešta ih u listu.*
2. *Napisati funkciju koja u jednom prolazu kroz zadatu listu celih brojeva pronalazi maximalan strogo rastući podniz.*
3. *Koristeći funkcije pod a) i b) napisati program koji u datoteku* `Rezultat.txt` *upisuje nađeni strogo rastući podniz.*

Zadatak 65 Jun, 2004. IIgrupa *Imena dve datoteke koje sadrže cele brojeve unose se kao argumenti komandne linije.*

1. *Napisati funkciju koja iz datoteke u ˇcitava brojeve i smešta ih u rastuće uredjenu listu (listu čiji su elementi poredjani u rastućem poretku).*
2. *Napisati funkciju koja od brojeva dve rastući uredjene liste formira treću koja je takodje rastući uredjena.*
3. *Koristeći funkcije pod a) i b) napisati program koji sortira brojeve (u rastućem poretku) koji se nalaze u datotekama čija su imena argumenti komandne linije i upisuje ih u datoteku* `Rezultat.txt`.

Zadatak 66 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 67 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 68 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 69 Drugi kolokvijum za II tok 2004.godine - rad na računaru *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 70 Drugi kolokvijum za II tok 2004.godine - rad na računaru *Napisati program koji iz tekstualne datoteke čiji je put dat u argumentu komandne linije učitava različite prirodne brojeve i:*

1. *dodaje ih redom u uredjeno binarno stablo*

2. *u dobijenom drvetu izračunava dužinu najdužeg puta od korena do nekog lista i*
3. *štampa u rastućem poretku (bez ponavljanja) sve brojeve koji su nalaze na putevima te dužine od korena do listova.*

Zadatak 71 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 72 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 73 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 74 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 75 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 76 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 77 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 78 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 79 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 80 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 81 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 82 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 83 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 84 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 85 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 86 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 87 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 88 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 89 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 90 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 91 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 92 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 93 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 94 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 95 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 96 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 97 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 98 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 99 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 100 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 101 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 102 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 103 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 104 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.*