Programiranje 2

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

May 28, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Bit-operatori

!!!Ne mešati sa logičkim operatorima!!!

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

Primer 1 *Demonstracija bitskih operatora*

```
#include <stdio.h>

main()
{
   printf("%o %o\n",255,15); /*255 i 15 se ispisuju u oktalnom zapisu*/
   printf( "255 & 15 = %d\n", 255 & 15 );
   printf( "255 | 15 = %d\n", 255 | 15 );
   printf( "255 ^ 15 = %d\n", 255 ^ 15 );
   printf( "2 << 2   = %d\n", 2 << 2 );
   printf( "16 >> 2  = %d\n", 16 >> 2 );
}

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

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

Primer 2 *print_bits - stampa bitove u zapisu datog celog broja x.*

```
#include <stdio.h>

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

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

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

    putchar('\n');
}


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

Izlaz iz programa:
00000000000000000000000001111111
00000000000000000000000010000000
00000000111111110000000011111111
11111111111111111111111111111111
```

Primer 3 *Program proverava da li se na k-tom mestu nalazi 1.*

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

Primer 4 *Program postavlja na k-to mesto 1*

```
#include <stdio.h>

void print_bits(int x);

main(){
  int n,k;
  printf("Unesite broj i poziciju tog broja koju zelite da proverite:\n");
  scanf("%d %d",&n,&k);
  printf("Binarno, une\v seni broj je\n");
  print_bits(n);
  printf("Novi broj je %d\n",(n |(1<<k)));
  printf("Binarno, novi broj je\n");
  print_bits((n |(1<<k)));
  return 0;
}
```

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

VAŽNO:

Vrednost izraza a<<b je jednaka vrednosti $a * 2^b$.

Vrednost izraza a>>b je jednaka vrednosti $a/2^b$.

Bitovski operatori su efikasniji od operacija * i / tako da svaki put kad se javi potreba za množenjem ili deljenjem stepenom broja dva, treba koristiti operacije << i >>!!!

Primer 5 *sum_of_bits - izračunava sumu bitova datog neoznačenog broja.*

```
#include <stdio.h>

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

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

    putchar('\n');
}

/*
int sum_of_bits(unsigned x)
{

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

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

    return br;
}

*/

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

    return br;
}


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

Primer 6 *Funkcija koja broji bitove postavljene na 1 u broju*

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

Primer 7 *get_bits, set_bits, invert_bits - izdvajanje, postavljanje i invertovanje pojedinacnih bitova*

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

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

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

    putchar('\n');
}


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

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

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

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

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

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

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

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

}
```

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

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

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

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

Izlaz iz programa:
00001010101000001010111110100000
00000000000000000000000010101111
00001010101000001111111110100000
00001010101000000101000010100000
```

Primer 8 *right_rotate_bits, mirror_bits - rotiranje i simetrija bitova.*

```
#include <stdio.h>

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

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

    putchar('\n');
}


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

    /* Postupak se ponavlja n puta */
    for (i = 0; i < n; i++)
    {
```

```
        /* Poslednji bit broja x */
        unsigned last_bit = x & 1;

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

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

        x |= last_bit<<wl-1;
    }

    return x;
}

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

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

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

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

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

Izlaz iz programa:
11111010111100001111101011110000
00001111010111110000111101011111
00111110101111000011111010111100
```

Zadaci za vežbu:

Zadatak 1 *(Ispitni zadatak, februar 2007.) Sastaviti funkciju koja za dati ceo broj tipa unsigned*

long int vraća

(a) najmanji co broj koji se sastoji od istog broja 0 i 1 i koji je tipa unsigned long

(b) najveći co broj koji se sastoji od istog broja 0 i 1 i koji je tipa unsigned long

Zadatak 2 *U uniju staviti long i float i pročitati mantisu i eksponent.*

Zadatak 3 *Napisati funkciju koja vraća kao rezultat broj koji se dobija od broja x tipa unsigned tako što mu se sačuvaju n krajnjih desnih bitova, a ostali se postave na nulu.*

Zadatak 4 *Napisati funkciju koja vraća kao rezultat broj koji se dobija od broja x tipa unsigned tako što mu očistiti n bitova (postaviti nule) počev od pozicije p.*

Zadatak 5 *Napisati funkciju koja vraća kao rezultat broj koji se dobija od broja x tipa unsigned tako što mu invertuje (prevesti jedan u nulu i nulu u jedinicu) n bitova počev od pozicije p.*

Zadatak 6 *Napisati funkciju koja vrsi rotaciju neoznacenog broja x za n pozicija u levo.*

2

Programski jezik C

[1]

2.1 Argumenti komandne linije

Primer 9 *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 10 *Program ispisuje opcije navedene u komandnoj liniji.* K&R *rešenje.*

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

```
/* 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 11 *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 12 *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;
}
```

2.2 Polinomi

Primer 13 *Program ilustruje rad sa polinomima.*

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

#define max(a, b) ((a) > (b) ? (a) : (b))

/*
*   Polinom 3*x^3 + 2*x + 1.5 se predstavlja kao:
*   stepen -> 3
```

```
*   koeficijenti -> 1.5 2 0 3 x x x x x ... x
*/
typedef struct polinom {
    float koeficijenti[21];
    int stepen;
} Polinom;

/*
*   Funkcija vraca koeficijent uz x^i u polinomu p
*   Zbog efikasnosti ne prenosimo celu strukturu vec
*   samo pokazivac na strukturu.
*/
float vratiKoeficijent(Polinom* p, int i) {
    return i <= p->stepen ? p->koeficijenti[i] : 0.0f;
}

/*
*   Funkcija postavlja koeficijent uz x^i u polinomu p na
*   dati koeficijent k
*/
void postaviKoeficijent(Polinom* p, int i, float k)
{
    int j;
    /*
       Ukoliko je stepen polinoma bio manji, postavljamo
           sve koeficijente izmedju na 0.0 i uvecavamo stepen
        */
    if (i > p->stepen) {
        for (j = p->stepen+1; j < i; j++)
            p->koeficijenti[j] = 0.0f;
        p->stepen = i;
    }

    p->koeficijenti[i] = k;
}

/*
*    Funkcija kreira polinom datog stepena sa datim nizom
*    koeficijenata. Pretpostavlja se da su koeficijenti
*    u datom nizu koeficijenti[] uredjeni opadajuce po stepenima
*    polinoma.
*/
Polinom napraviPolinom(int stepen, float koeficijenti[]) {
    int i;
    Polinom p;
    p.stepen = stepen;
    for (i = 0; i <= stepen; i++) {
        postaviKoeficijent(&p, i, koeficijenti[stepen - i]);
    }
    return p;
}
```

```
/*
*   Funkcija ispisuje polinom u citljivijem obliku.
*   Npr: 3.0*x^3 + 0.0*x^2 + 2.0*x^1 + 1.5*x^0
*/
void ispisiPolinom(Polinom* p) {
    int i;
    for (i = p->stepen; i >= 0; i--) {
        printf("%.2f*x^%d", vratiKoeficijent(p, i), i);
        if (i > 0)
            printf(" + ");
    }
    printf("\n");
}

/*
*   Funkcija izracunava vrednost polinoma u tacki x
*   Hornerovom shemom. Npr.
*   3*x^3 + 2*x + 1.5   =
*   (((0*x + 3)*x + 0)*x + 2)*x + 1.5
*   Postupak izracunavanja p(10):
*   0.0
*   10 * 0.0 + 3 = 3.0
*   10 * 3.0 + 0 = 30.0
*   10 * 30.0 + 2 = 302.0
*   10 * 302.0 + 1.5 = 3021.5
*/
float vrednost(Polinom* p, float x) {
    int i;
    float suma = 0.0f;
    for (i = p->stepen; i >= 0; i--) {
        suma = suma*x + vratiKoeficijent(p, i);
    }
    return suma;
}

/*
*   Funkcija sabira dva polinoma
*/
Polinom saberi(Polinom* p, Polinom* q) {
    int i;
    Polinom zbir;
    zbir.stepen = max(p->stepen, q->stepen);
    for (i = 0; i <= zbir.stepen; i++)
        postaviKoeficijent(&zbir, i,
            vratiKoeficijent(p, i) + vratiKoeficijent(q, i));
    return zbir;
}

/*
*   Funkcija mnozi dva polinoma. Npr.
*
*   1*x^2 + 2*x + 3
```

```
*   4*x + 7
*
*   0.0*x^3 + 0.0*x^2 + 0.0*x + 0.0
*   0.0*x^3 + 0.0*x^2 + 0.0*x + 21.0    i=0 j=0
*   0.0*x^3 + 0.0*x^2 + 12.0*x + 21.0   i=0 j=1
*   0.0*x^3 + 0.0*x^2 + 26.0*x + 21.0   i=1 j=0
*   0.0*x^3 + 8.0*x^2 + 26.0*x + 21.0   i=1 j=1
*   0.0*x^3 + 15.0*x^2 + 26.0*x + 21.0  i=2 j=0
*   4.0*x^3 + 15.0*x^2 + 26.0*x + 21.0  i=2 j=1
*/

Polinom pomnozi(Polinom* p, Polinom* q) {
    int i, j;
    Polinom proizvod;
    proizvod.stepen = p->stepen + q->stepen;
    for (i = 0; i <= proizvod.stepen; i++) {
        postaviKoeficijent(&proizvod, i, 0.0f);
    }

    for (i = 0; i <= p->stepen; i++) {
        for (j = 0; j <= q->stepen; j++) {
            /* r[i+j] = r[i+j] + p[i]*q[j] */
            postaviKoeficijent(&proizvod, i+j,
                vratiKoeficijent(&proizvod, i+j) +
                vratiKoeficijent(p, i) *
                vratiKoeficijent(q, j));
        }
    }
    return proizvod;
}

main() {
    float pkoeficijenti[] = {1.0f, 2.0f, 1.0f};
    Polinom p = napraviPolinom(2, pkoeficijenti);
    float qkoeficijenti[] = {1.0f, 1.0f};
    Polinom q = napraviPolinom(1, qkoeficijenti);
    Polinom r = pomnozi(&p, &q);
    ispisiPolinom(&r);
}
```

2.3 Rad sa velikim celim brojevima

Primer 14 *Program ilustruje rad sa velikim celim brojevima. Brojevi se interno reprezentuju preko niza svojih cifara.*

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

/* Funkcija obrce cifre prosledjenog niza */
void obrni_cifre (int cifre[], int duzina)
{
```

```
    int i, j;
    for (i=0, j=duzina-1; i<j; i++, j--)
    {
        int pom = cifre[i];
        cifre[i] = cifre[j];
        cifre[j] = pom;
    }
}

/* Funkcija sa standardnog ulaza ucitava niz cifara, duzine
   najvise max_cifre i smesta ga u niz brojeva cifre[]. Zatim
   se niz cifre[] obrce kako bi cifre najmanje tezine bile
   na pocetku niza.
   Kao rezultat, funkcija vraca broj cifara ucitanog broja */
int uzmi_broj (int cifre[], int max_cifre)
{
    int duzina = 0;
    char c;

    while ( --max_cifre>0 && isdigit(c=getchar()))
        cifre[duzina++]=c-'0';

    obrni_cifre(cifre, duzina);

    return duzina;
}


/* Funkcija ispisuje "veliki" broj predstavljen
   nizom cifara cifre, duzine duzina, na standardni
   izlaz imajuci u vidu da su cifre u nizu zapisane
   "naopako" tj. pocevsi od cifre najmanje tezine */
void ispisi_broj(int cifre[],int duzina)
{
    int i;
    for (i=duzina-1; i>=0; i--)
        printf("%d",cifre[i]);
    putchar('\n');
}


/* Da li su dva broja data svojim nizovima cifara i duzinama jednaka?
   Funkcija vraca 1 ako jesu, a 0 ako nisu */
int jednaki(int a[], int duzina_a, int b[], int duzina_b)
{
    int i;

    /* Poredimo duzine */
    if (duzina_a != duzina_b)
        return 0;

    /* Ako su brojevi iste duzine, poredimo cifru po cifru
```

```
       pocevsi od pozicije najvece tezine */
    for (i=0; i<duzina_a; i++)
        if (a[i] != b[i])
            return 0;

    return 1;
}


/* Funckija poredi dva broja a i b, data svojim nizovima cifara i duzinama
   i vraca:
      1  ako je a>b
      0  ako je a=b
      -1 ako je b>a
*/
int uporedi(int a[], int duzina_a, int b[], int duzina_b)
{
    int i;
    /* Uporedjujemo duzine brojeva a i b */
    if (duzina_a > duzina_b)
        return 1;
    if (duzina_a < duzina_b)
        return -1;

    /* U ovom trenutku znamo da su brojevi iste duzine, tako da
       prelazimo na poredjenje cifre po cifre, pocevsi od cifre
       najvece tezine */
    for (i=duzina_a-1; i>=0; i--)
    {
        if (a[i] > b[i])
            return 1;
        if (a[i] < b[i])
            return -1;
    }

    return 0;
}

/* Funkcija sabira dva broja data svojim nizovima
   cifara i duzinama i rezultat ispisuje na ekran
 */
void saberi(int a[], int duzina_a, int b[], int duzina_b)
{
    /* Rezultat, zadan svojim nizom cifara i duzinom */
    int rezultat[MAX_CIFRE];
    int duzina_rezultata;
    int i;
    /* Prenos sa prethodne pozicije */
    int prenos = 0;

    /* Sabiranje vrsimo dok ne prodjemo sve cifre duzeg od brojeva a i b */
    for(i=0; i<duzina_a || i<duzina_b; i++)
```

```
    {
        /* Sabiramo i-tu cifra broja a (ako postoji) sa i-tom cifrom
           broja b (ako postoji) i prenos sa prethodne pozicije */
        int cifra_rezulata=((i<duzina_a)? a[i] : 0) +
                    ((i<duzina_b)? b[i] : 0) +
                    prenos;

            /* Nova cifra rezultata */
        rezultat[i] = cifra_rezulata%10;

        /* Prenos na sledecu poziciju */
            prenos = cifra_rezulata/10;
    }

    /* Kada smo zavrsili sa svim ciframa brojeva a i b moguce je da je
    postojao prenos na sledecu poziciju u kom slucaju uvecavamo duzinu
    rezultata. Inace duzinu rezultata postavljamo na duzinu duzeg od
    brojeva a i b, koja se nalazi trenutno u  promenjivoj i */
    if (prenos != 0)
    {
    if (i>=MAX_CIFRE)
        printf("Doslo je do prekoracenja!\n");
    else
    {
        rezultat[i] = prenos;
        duzina_rezultata = i+1;
       }
    }
    else
    duzina_rezultata=i;

    /* Ispisujemo rezultat */
    ispisi_broj(rezultat, duzina_rezultata);
}

/* Funkcija mnozi broj, dat svojim nizom cifara i duzinom, datom cifrom c
   i rezultat ispisuje */
void pomnozi_cifrom (int a[], int duzina_a, int cifra)
{
    /* Rezultat, zadan svojim nizom cifara i duzinom */
    int rezultat[MAX_CIFRE];
    int duzina_rezultat;

    int i, prenos = 0;
    for (i=0; i<duzina_a; i++)
    {
        /* Svaku cifru broja a mnozimo cifrom c, dodajemo na to
           prenos sa prethodne pozicije i to smestamo u promenljivu
           pom */
        int pom  = a[i]*cifra + prenos;

        /* Nova cifra rezultata */
```

```
        rezultat[i] = pom%10;
        /* Prenos na sledecu poziciju */
        prenos = pom/10;
    }

    /* Kada smo zavrsili sa svim ciframa broja a, moguce je da je
       postojao prenos na sledecu poziciju u kom slucaju uvecavamo
       duzinu rezultata. Inace duzinu rezultata postavljamo na duzinu
       broja a, koja se nalazi trenutno u promenjivoj i */
    if (prenos != 0)
    {
        if (i>=MAX_CIFRE)
            printf("Doslo je do prekoracenja !\n");
        else
        {
            rezultat[i] = prenos;
            duzina_rezultata = duzina_a+1;
        }
    }
    else
        duzina_rezultata = duzina_a;

   /* Ispisujemo rezultat */
   ispisi_broj(rezultat, duzina_rezultata);
}


/* Funkcija mnozi dva broja data svojim nizovima cifara i duzinama i proizvod
   ispisuje na standardni izlaz */
void pomnozi (int a[], int duzina_a, int b[], int duzina_b)
{
    /* Ova funkcija se gradi kombinovanjem algoritama mnozenja broja cifrom i
       sabiranja dva broja */

    int i,j,k;

    /* Broj pom ce da sadrzi rezultat mnozenja broja a jednom po jednom cifrom broja b,
       Dok ce na broj rezultat da se dodaje svaki put (10^i)*pom */
    int pom[MAX_CIFRE], duzina_pom;
    int rezultat[MAX_CIFRE], duzina_rezultata;


    duzina_rezultata = 0;

    /* Za svaku cifru broja a */
    for (i=0; i<duzina_a; i++)
    {
        /* vrsimo mnozenje broja b i-tom cifrom broja a */
        int prenos = 0;
        for (j=0; j<duzina_b; j++)
        {
            int pm = b[j]*a[i] + prenos;
```

```
            pom[j] = pm%10;
            prenos = pm/10;
        }

        if (prenos)
        {
            pom[j] = prenos;
            duzina_pom = j+1;
            }
        else
            duzina_pom = j;


            /* Zatim dodajemo broj pom na rezultat, ali dodavanje pocinjemo od i-te cifre rezultata.
        prenos = 0;
        for (k=0; k<duzina_pom || i+k<duzina_rezultata; k++)
            {
            int pm=((i+k<duzina_rezultata) ? rezultat[i+k] : 0) + pom[k] + prenos;
            rezultat[i+k] = pm%10;
            prenos = pm/10;
        }

            if (prenos)
        {
            rezultat[i+k] = prenos;
            duzina_rezultata = i+k+1;
            }
        else
            duzina_rezultata = i+k;

    }

    /* Ispisujemo rezultat */
    ispisi_broj(rezultat, duzina_rezultata);

}


/* Primer koriscenja funkcija */
int main()
{
    /* duzina brojeva a i b */
    int duzina_a, duzina_b;

    /* nizovi cifara brojeva a i b */
    int a[MAX_CIFRE], b[MAX_CIFRE];

    /* Ucitavaju se brojevi */
    printf("Unesite prvi broj : ");
        duzina_a = uzmi_broj(a,MAX_CIFRE);
        printf("Unesite drugi broj : ");
    duzina_b = uzmi_broj(b, MAX_CIFRE);
```

```
    /* Sabiraju se i ispisuje se zbir */
    printf("Zbir je : ");
    saberi(a, duzina_a, b, duzina_b);


    /* Mnoze se i ispisuje se proizvod */
    printf("Proizvod je : ");
        pomnozi(a, duzina_a, b, duzina_b);

    return 0;
}
```

Zadaci za vežbu:

Zadatak 7 *Ilustracija rada sa argumentima komandne linije (korišćenje naredbe switch u programu).*

Zadatak 8 *(Ispitni zadatak, februar 2007.) Sastaviti program koji ispisuje prvih 100 elemenata Fibonačijevog niza zadatog sledećim formulama*

$f_1 = 1$, $f_2 = 2$, $f_n = f_{n-1} + f_{n-2} (n > 2)$

tačno u svakoj cifri. (Napomena: ako je f_n tipa unsigned long moguće je ispisati prva 43 elementa niza.)

Zadatak 9 *Napisati funkciju koja izračunava faktorijel broja tačno u svakoj cifri.*

3

Programski jezik C

[1]

3.1 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 15 *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/~filip`

```
      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 16 *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 17 *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 18 *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 19 *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;

}
```

3.2 Linearna i binarna pretraga niza

Primer 20 *Linearno pretraživanje*

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

main()
{
    /* Inicijalizacija niza moguca je
    na ovaj nacin*/
    int a[] = {4, 3, 2, 6, 7, 9, 11};
    /* Da bi smo odredili koliko clanova
    ima niz mozemo koristiti operator
    sizeof*/
    int br_elem = sizeof(a)/sizeof(int);
    int x;
    int i;
    printf("Unesite broj koji trazimo : ");
    scanf("%d",&x);
    i = linearna_pretraga(a, br_elem, x);
    if (i == -1)
        printf("Element %d nije nadjen\n",x);
    else
        printf("Element %d je nadjen na poziciji %d\n",x, i);
}
```

Primer 21 *Binarna pretraga niza*

```
/* Binarna pretraga niza celih brojeva - iterativna verzija*/

#include <stdio.h>

/* Funkcija proverava da li se element x javlja unutar niza
celih brojeva a.
Funkcija vraca poziciju na kojoj je element nadjen odnosno
-1 ako ga nema.
!!!!! VAZNO !!!!!
Pretpostavka je da je niz a uredjen po velicini
*/
int binarna_pretraga(int a[], int n, int x)
```

```
{
    /* Pretrazujemo interval [l, d] */
    int l = 0;
    int d = n-1;
    /* Sve dok interval [l, d] nije prazan */
    while (l <= d)
    {
        /* Srednja pozicija intervala [l, d] */
        int s = (l+d)/2;
        /* Ispitujemo odnos x i a[srednjeg elementa] */
        if (x == a[s])
            /* Element je pronadjen */
            return s;
        else if (x < a[s])
                {
                 /* Pretrazujemo interval [l, s-1] */
                 d = s-1;
                }
            else
            {
                /* Pretrazujemo interval [s+1, d] */
                l = s+1;
            }
    }
    /* Element nije nadjen */
    return -1;
}

main()
{
    int a[] = {3, 5, 7, 9, 11, 13, 15};
    int x;
    int i;
    printf("Unesi element koji trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, sizeof(a)/sizeof(int), x);
    if (i==-1)
        printf("Elementa %d nema\n", x);
    else
        printf("Pronadjen na poziciji %d\n", i);
}
```

Zadaci za vežbu:(Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~milan`)

Zadatak 10 *a) Napisati funkcije za sortiranje niza u rastućem poretku za razne tipove elemenata (float,char, long) i raznim metodama (select, insert, bubble). Na primer, prototip funkcije za sortiranje niza celih brojeva metodom select sorta može biti:*

`void int_select_sort(int a[], int n)`

Objedniti sve te funkcije sortiranja u jedan fajl "sort.c" (bez main funkcije), a zatim kreirati zaglavlje "sort.h" koje će sadržati deklaracije (prototipove) svih napisanih funkcija za sortiranje.

b) Napisati funkcije koje generišu niz slučajnih brojeva različitih tipova (float,char, long) dužine n, iz opsega [l, d].

Prototip funkcije za generisanje niza od n celih brojeva iz opsega [l,d] može biti:

```
void random_int_niz(int a[], int n, int l, int d)
```

Napomena: Broj celobrojnog tipa (npr. tipa long) iz opsega [l, d] se može generisati sledećim izrazom:

```
a= l + (long)rand() % (d-l);
```

Slično, broj realnog tipa (npr. float) iz opsega [l, d] se može generisati sledećim izrazom:

```
a= l + ((float)rand()/RAND_MAX) *(d-l);
```

Inicijalizacija početnih vrednosti (semena) algoritma za generisanje slučajnih brojeva ostvaruje se pozivom funkcije:

```
srand(time(NULL));
```

Objedniti sve te u jedan fajl "random.c", a zatim kreirati zaglavlje random.h koje će sadržati deklaracije (prototipove) svih napisanih funkcija za generisanje slučajnih brojeva.

c) U fajlu "main.c" napisati main() funkciju, uz uključenje zaglavlja "sort.h" i "random.h" direktivom `#include`*. U okviru funkcije main(), generisati niz od 15 slučajnih brojeva u intervalu od -10000 do 10000, sortirati nekom od funkcija i ispisati tako dobijeni sortirani niz.*

d) Odvojeno prevesti svaki fajl sa izvornim kodom. Na kraju sve dobijene objektne fajlove povezati u jedan izvršni fajl sort.

Uputstvo: Program prevesti naredbama:

```
gcc -c -o sort.o sort.c
gcc -c -o random.o random.c
gcc -c -o main.o main.c
gcc -o sort main.o sort.o random.o
```

e) Proširiti funkciju main() iz prethodnog primera tako da omogućava korisniku izbor da li želi da ručno unese brojeve za sortiranje, ili želi da se niz date dužine generiše na slučajan način. Ovo se na primer može uraditi argumentima komandne linije: npr. ako se program pozove sa:

```
./sort -r 10
```

tada ce se na slučajan način generisati niz od deset brojeva, dok ako se navede:

```
./sort -i
```

tada se od korisnika očekuje da brojeve unosi na standardnom ulazu.

Implementirati i opciju -c:

```
./sort -c 2 -1 5 67 2
```

kojom se specificira da se niz unosi sa komandne linije nakon opcije -c.

f) Prethodni program obogatiti još i mogućnošću rada sa skupovima predstavljenim preko sortiranih nizova. Napisati funkcije za izračunavanje unije, preseka, razlike i pripadanja, objediniti ih u fajl "skup.c" a zatim kreirati zaglavlje "skup.h".

4

Programski jezik C

[1]

4.1 Rekurzija

Rekurzija je postupak rešavanja zadataka, u kome neka funkcija (direktno ili indirektno) poziva samu sebe. Pri tome se vodi računa da se svaki dublji poziv izvršava za jednostavniji problem od polaznog, a da se najjednostavniji problemi rešavaju direktno, tj. bez dalje upotrebe rekurzije (jednostavnost problema se definiše na pogodan način, prema prirodi samog problema, a najčešće kao veličina ulaznog argumenta rekurzivne funkcije).

Pri rešavanju zadatka, rekurziju je najbolje shvatiti i koristiti na sledeći način: Potrebno je da umemo da neposredno rešimo najjednostavniji slučaj datog problema, i da složenije slučajeve svedemo na jednostavnije, tj. da ih rešimo koristeći jednostavnije slučajeve. Pri tome ne treba da brinemo o tome kako će biti rešeni ti jednostavniji slučajevi na koje se složeni slučaj svodi - tu rekurzija radi za nas.

Primer 22 *Rekurzivna i iterativna varijanta funkcije koja stepenuje ceo broj na celobrojni pozitivan izlozilac.*

```
#include <stdio.h>

int stepen(int x, int k)
{

    printf("Racunam stepen(%d, %d)\n",x,k); /* Ilustracije radi! */

    if (k == 0)
        return 1;
    else
        return x*stepen(x, k-1);


    /* Krace se moze zapisati i kao

    return k==0 ? 1 : x*stepen(x, k-1); */
```

[1] Zasnovano na primerima sa sajta `http://www.matf.bg.ac.yu/~filip`, i zbirke zadataka "Programiranje i programiranje", Milan Vugdelija

```
}

/* Druga verzija prethodne funkcije.
Obratiti paznju na efikasnost u odnosu na prvu verziju!*/
int stepen2(int x, int k)
{
    printf("Racunam stepen2(%d, %d)\n",x,k); /* Ilustracije radi! */
    if (k == 1)
        return x;
    else
         /* Ako je stepen paran*/
        if((k%2)==0)
            return stepen2(x*x, k/2);
        else
            return x * stepen2(x*x, k/2);


}

/* Iterativna verzija prethodne funkcije */
int stepen_iterativno(int x, int k)
{
    int i;
    int s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}


/* Iterativna verzija prethodne funkcije */
int stepen_iterativno(int x, int k)
{
    int i;
    int s = 1;
    for (i = 0; i<k; i++)
        s*=x;

    return s;
}


main()
{
    printf("%d", stepen2(2, 8));
    printf("\n------------------\n");
    printf("%d", stepen(2, 8));
}

/* Izlaz iz programa:
```

```
Racunam stepen2(2, 8)
Racunam stepen2(4, 4)
Racunam stepen2(16, 2)
Racunam stepen2(256, 1)
256
------------------
Racunam stepen(2, 8)
Racunam stepen(2, 7)
Racunam stepen(2, 6)
Racunam stepen(2, 5)
Racunam stepen(2, 4)
Racunam stepen(2, 3)
Racunam stepen(2, 2)
Racunam stepen(2, 1)
Racunam stepen(2, 0)
256
*/
```

Primer 23 *Rekurzivna i iterativna varijanta računanja Fibonačijevih brojeva.*

```
/* Fibonacijev niz brojeva se definise kao niz u kome su prva dva broja
   jednaka 1, a za ostale brojeve vazi da je svaki sledeci zbir dva
   prethodna. tj.
    f(0) = 1, f(1) = 1, f(n) = f(n-1) + f(n-2)
*/

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

/* Rekurzivna implementacija - obratiti paznju na neefikasnost funkcije */
int Fib(int n)
{
    printf("Racunam Fib(%d)\n",n); /* Ilustracije radi! */

    if((n==0)||(n==1))
        return 1;
    else
        return(Fib(n-1)+Fib(n-2));

    /* Krace moze kao:
    return (n == 0 || n == 1) ? 1 : Fib(n-1) + Fib(n-2); */
}


/* Iterativna verzija bez niza */
int Fib_iterativno(int n)
{
    /* Promenljiva pp cuva pretposlednji, a p poslednji element niza */
    int pp = 1, p = 1;
    int i;
    for (i = 0; i <= n-2; i++)
```

```
    {
        int pom = pp;
        pp = p;
        p = p + pom;
    }
    return p;
}


main()
{
    printf("Fib(5) = %d\n", Fib(5));

}

/* Izlaz iz programa:
Racunam Fib(5)
Racunam Fib(4)
Racunam Fib(3)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(1)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(3)
Racunam Fib(2)
Racunam Fib(1)
Racunam Fib(0)
Racunam Fib(1)
Fib(5) = 8
*/
```

Primer 24 *Rekurzivna i iterativna varijanta računanja faktorijela prirodnog broja.*

```
unsigned long faktorial(int n)
{

    if (n == 1)
        return 1;
    else
        return n*faktorial(n-1);


    /* Krace moze kao:
    return n == 1 ? 1 : n*faktorial(n-1); */
}

/* Iterativna verzija prethodne funkcije */
unsigned long faktorial_iterativno(int n)
{
```

```
    long f = 1;
    int i;
    for (i = 1; i<=n; i++)
        f *= i;
    return f;
}

main()
{
    printf("5! = %d\n", faktorial(5));
}
```

Primer 25 *Rekurzivna i iterativna varijanta računanja sume niza celih brojeva.*

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

/* Krace moze kao:
return n == 1 ? a[0] : suma_niza(a, n-1)+a[n-1];*/
}

/* Iterativna verzija prethodne funkcije */
int suma_niza_iterativno(int a[], int n)
{
int suma = 0;
int i;
for (i = 0; i<n; i++)
suma+=a[i];
return suma;
}
```

Primer 26 *Štampanje celog broja.*

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

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

Primer 27 *Rekurzivna varijanta binarne pretrage niza celih brojeva.*

```
#include <stdio.h>

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

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

int binarna_pretraga(int a[], int l, int d, int x)
{
    /* Ukoliko je interval prazan, elementa nema */
    if (l > d)
        return -1;


    /* Srednja pozicija intervala [l, d] */
    int s = (l+d)/2;

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

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

    printf("Unesi element kojega trazimo : ");
    scanf("%d",&x);
    i = binarna_pretraga(a, 0, sizeof(a)/sizeof(int)-1, x);
```

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

Primer 28 *Sortiranje niza celih brojeva - QuickSort algoritam*

```
#include<stdio.h>

/* Funkcija menja mesto i-tom i j-tom elementu niza a */
void razmeni(int a[], int i, int j)
{
    int pom = a[i];
    a[i] = a[j];
    a[j] = pom;
}

/* Funkcija sortira deo niza brojeva a izmedju pozicija l i d*/
void quick_sort(int a[], int l, int d)
{
    int i, poslednji, pivot;

    /* Ukoliko je interval [l, d] prazan nema nista da se radi */
    if (l >= d)
        return;

     /* Srednji element uzimamo za pivot i postavljamo ga na pocetak */
    razmeni(a, l, (l + d)/2);

    /* Niz organizujemo tako da pivot postavimo na pocetak, zatim da iza
       njega budu svi elementi manji od njega, pa zatim svi elementi koji
       su veci od njega  tj. pivot < < < < > > > >*/

    /* poslednji je pozicija poslednjeg elementa niza za koji znamo da je
       manji od pivota. U pocetku takvih elemenata nema */
    poslednji = l;
    for (i = l+1; i <= d; i++)
        /* Ukoliko je element na poziciji i manji od pivota,
           postavljamo ga iza niza elemenata manjih od pivota,
           menjajuci mu mesto sa prvim sledecim */
        if (a[i] < a[l])
            razmeni(a, ++poslednji, i);

    /* Zahtevanu konfiguraciju < < < < piv > > > > dobijamo tako
       sto zamenimo mesto pivotu i poslednjem elementu manjem od njega */
        razmeni(a, l, poslednji);

    /* Pivot se sada nalazi na poziciji poslednji */
    pivot = poslednji;

    /* Rekurzivno sortiramo elemente manje od pivota */
```

```
    quick_sort(a, l, pivot-1);
    /* Rekurzivno sortiramo elemente vece pivota */
    quick_sort(a, pivot+1, d);
}


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

    quick_sort(a, 0, n-1);

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

}
```

Zadaci za vežbu:

Zadatak 11 *Napisati program koji upotrebom rekurzivne funkcije ispituje da li je dati string palindrom.*

Zadatak 12 *Za dati broj pomoću rekurzivne funkcije ispisati broj koji se piše istim ciframa ali u obrnutom poretku.*

Zadatak 13 *Napisati rekurzivnu funkciju koja vraća odgovor na pitanje da li je broj cifara njenog argumenta paran.*

Zadatak 14 *Napisati rekurzivnu funkciju koja računa zbir cifara na neparnim pozicijama, računajući skraja (sdesna nalevo).*

5

Programski jezik C

[1]

5.1 Pokazivači na funkcije

Primer 29 *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 (*f) (int) u argumentu funkcije sumiraj je pokazivac na funkciju sa imenom f,
   koja kao argument prima promenljivu tipa int i vraca kao rezultat vrednost tipa int */

int sumiraj(int (*f) (int), int n) {
    int i, suma=0;
    for (i=1; i<=n; i++)
        suma += (*f)(i);

    return suma;
}

main(){

/*  U pozivu funkcije sumiraj, kvadrat, kub i parni_broj su adrese funkcija pa operator & nije
    neophodan, iz istog razloga zbog kojeg on nije bio potreban ni za ime niza.*/

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));
}
```

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

```
 /*
 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
 */
```

Podsetimo se sada pokazivača na tip void:

```
void *pp;
```

Svaki pokazivač se može pretvoriti u tip void * i opet vratiti u svoj prvobitni tip bez gubitka informacije. Njemu se dakle može dodeliti 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 30 *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

*/
```

5.2 qsort – funkcija iz standardne biblioteke

Prototip funkcije qsort iz stdlib.h je:

```
void qsort(void *niz, int duzina_niza, int velicina_elementa_niza,
    int (*poredi)(const void*, const void*) )
```

Ova funkcija sortira niz niz[0], niz[1],...,niz[duzina_niza - 1] elemenata veličine velicina_elementa_niza. Funkcija poredi vrši poređenje dva elementa niza, vraća pozitivnu vrednost ako je prvi element veći od drugog, 0 ako su jednaki i negativnu vrednost ako je prvi manji. Tada će se niz sortirati u

rastućem poretku. Modifikacijom ove funkcije niz se može sortirati u opadajućem poretku (ukoliko vraća pozitivnu vrednost ako je prvi manji, 0 ako su jednaki i negativnu vrednost ako je prvi veći).

Primer 31 *Upotrebom qsort funkcije iz standardne biblioteke izvršiti sortiranje niza celih i niza realnih 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);
    printf("Sortirani niz celih brojeva je:\n");
    for(i=0; i<n; i++)
        printf("%d\t",niz[i]);

    printf("\n\nSortirani niz realnih brojeva je:\n");

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

}
```

Primer 32 *Izvršiti sortiranje niza reči, leksikografski odnosno po dužini, korišćenjem ugradjene qsort funkcije.*

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

/* Funkcija koja vrsi leksikografsko poredjenje dve reci.
   Vraca kao rezultat >0 ukoliko je prva rec veca, 0 ako su jednake
   i <0 ako je prva rec manja. Sortiranje ce biti u rastucem redosledu.*/
int poredi_leksikografski(const void* a, const void* b)
{    return strcmp(*(char**)a,*(char**)b);   }

/* Funkcija koja vrsi poredjenje po duzini dve reci, opadajuce!!!*/
int poredi_po_duzini(const void* a, const void* b)
{     return strlen(*(char**)b)-strlen(*(char**)a); }


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

    /* Sortiramo leksikografski i ispisujemo rezultat */
    qsort((void*)a, n, sizeof(char*), poredi_leksikografski);
    for (i=0; i<n; i++)
        printf("%s ", a[i]);
    putchar('\n');


    /* Sortiramo po duzini i ispisujemo rezultat */
    qsort((void*)a, n, sizeof(char*), poredi_po_duzini);
    for (i=0; i<n; i++)
        printf("%s ", a[i]);
    putchar('\n');
}
/*
Izlaz:
Dinja Jabuka Kruska Lubenica Sljiva
Lubenica Kruska Jabuka Sljiva Dinja
*/
```

5.3 bsearch – funkcija iz standardne biblioteke

Prototip funkcije qsort iz stdlib.h je:

```
void *bsearch (const void *kljuc, const void *niz, int duzina_niza, int velicina_elementa_niza,
    int (*poredi)(const void*, const void*) )
```

Ova funkcija u nizu niz[0], niz[1],...,niz[duzina_niza - 1] elemenata veličine velicina_elementa_niza traži element koji se poklapa sa *kljuc. Funkcija poredi mora vratiti vrednost ¿0 ako je prvi argument (ključ pretrage) veći od drugog argumenta (koji je element niza), 0 ako su jednaki i ¡0 ako je prvi manji. Elementi u nizu niz moraju biti u rastućem redosledu. Funkcija bsearch vraća pokazivač na pronađeni element ili NULL ako takav ne postoji.

Primer 33 *Binarno pretraživanje - korišćenje ugrađene bsearch funkcije.*

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

/* Vrsi se poredjenje podatka a po kome se pretrazuje niz sa elementom niza b.*/
int poredi(const void* a, const void *b)
{
    return *(int*)a-*(int*)b;
}

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

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

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

Primer 34 ***Binarna pretraga niza struktura - pretraga studenata po broju indeksa***

Datoteka sadrži podatke o uspehu studenata na kolokvijumima iz osnova programiranja. Prva linija datoteke sadrži broj studenata (¡1000), a zatim svaka sledeća linija sadrži ime i prezime odredjenog studenta, njegov broj indeksa (u obliku korisničkog imena na alas-u npr. mr01123) i broj poena na prvom i na drugom kolokvijumu. Redosled studenata u datoteci je odredjen na osnovu njihovog broja indeksa, i to tako da su na početku datoteke navedeni stariji studenti sa manjim brojevima indeksa (npr. mv02234 je ispred mn03123 jer je stariji, a mr03123 je ispred ml03234 jer ima manji broj indeksa).

a) Definisati strukturu podataka za čuvanje podataka o studentima

b) Napraviti funkciju koja poredi dva indeksa u skladu sa poretkom opisanim u uvodu zadatka

c) Napisati funkciju

int binary_search_stdlib(char indeks, ...)*

koja pozivom bibliotecke funkcije bsearch za dati indeks studenta vraća broj poena koje je ostvario na prvom plus drugom kolokvijumu.

d) Napisati program koji učitava podatke o studentima iz datoteke čije je ime navedeno kao argument komandne linije, zatim koristeći funkcije iz prethodnog dela zadatka ispisuje ukupan broj poena za svakog studenata čiji indeks korisnik unese sa standardnog ulaza, sve dok ne unese reč kraj za kraj.

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

/* Definicija strukture za cuvanje podataka o jednom studentu */
typedef struct _student
{
    char ime[15];
```

```
    char prezime[15];
    char indeks[8];
    int  kol_1;
    int  kol_2;
} student;

/* Studenti se smestaju u niz koji nema vi\v{s}e od 1000 elemenata.
   Pretpostavlja se da je niz sve vreme sortiran kako bi se
   moglo vrsiti binarno pretrazivanje */
student studenti[1000];
int br_stud;

/* Funkcija vrsi poredjenje indeksa. Manjim se smatraju stariji studenti
   sa manjim brojem indeksa. Funkca vraca negativnu vrednost ukoliko je
   prvi indeks manji, pozitivnu ukoliko je veci, a 0 ukoliko su indeksi
   jednaki. Informacije o smerovima se u potpunosti zanemaruju.
*/
int poredi_indekse(char* indeks_1, char* indeks_2)
{
    /* Izdvaja se informacija o godini upisa i
       broju indeksa za oba studenta*/
    char s_godina_1[3], s_godina_2[3];
    int  godina_1, godina_2;
    char s_broj_1[4], s_broj_2[4];
    int  broj_1, broj_2;


    /*mr99123 upisan je 1999 a mr01123 upisan je 2001*/

    /* Godina upisa je zapisana u 3 i 4 karakteru indeksa.*/
    strncpy(s_godina_1, indeks_1+2, 2);
    godina_1 = atoi(s_godina_1);
    if (godina_1 < 10) godina_1 += 100;
    godina_1 += 1900;

    strncpy(s_godina_2, indeks_2+2, 2);
    godina_2 = atoi(s_godina_2);
    if (godina_2 < 10) godina_2 += 100;
    godina_2 += 1900;

    /* Prvo poredimo godine */
    if (godina_1 < godina_2)
        return -1;

    if (godina_1 > godina_2)
        return 1;

    /* Ukoliko su godine jednake, izdvajamo brojeve indeksa
       koji su zapisani u 5. 6. i 7. karakteru indeksa */
    strncpy(s_broj_1, indeks_1+4, 3);
    broj_1 = atoi(s_broj_1);
    strncpy(s_broj_2, indeks_2+4, 3);
```

```
    broj_2 = atoi(s_broj_2);

    /* Umesto ovoga, prolazi i return strcmp(indeks_1+4, indeks_2+4); */

    /* Poredimo godine upisa */
    return broj_1-broj_2;
}


/* Funkcija poredjenja koja je potrebna prilikom poziva
   ugradjene funkcije bsearch.
   Vrsi se poredjenje prvog argumenta indeks koji je kljuc pretrage i tipa je string
   sa elementom niza koji je tipa struktura student */
int poredi(const void* indeks, const void* indeks_2)
{
    return poredi_indekse(*(char**)indeks, ((student*)indeks_2)->indeks);
}

int binary_search_stdlib(char* indeks)
{
    student *s = (student*)bsearch((const void*)&indeks, (const void*)studenti,
                                                br_stud, sizeof(student), poredi);
    return s==NULL ? -1 : s->kol_1+s->kol_2;
}

/* Glavni program */
main(int argc, char* argv[])
{
    FILE* datoteka;
    int i;
    char indeks[8];

    /* Proveravamo prisutnost argumenata komandne linije */
    if (argc<2)
    {   printf("Upotreba %s : ime_datoteke\n",argv[0]);
        return -1;
    }


    /* Otvaramo datoteku */
    if ((datoteka = fopen(argv[1], "r")) == NULL)
    {   printf("Greska prilikom otvaranja datoteke %s\n",argv[1]);
        return -1;
    }

    /* Ucivamo broj studenata i alociramo niz */
    fscanf(datoteka, "%d", &br_stud);

    /* Ucitavamo podatke o studentima */
    for (i = 0; i < br_stud; i++)
    {   fscanf(datoteka, "%s",studenti[i].ime);
        fscanf(datoteka, "%s",studenti[i].prezime);
```

```
        fscanf(datoteka, "%s",studenti[i].indeks);
        fscanf(datoteka, "%d", &studenti[i].kol_1);
        fscanf(datoteka, "%d", &studenti[i].kol_2);
    }

    /* Ispisujemo poene za svakog studenta kojeg korisnik trazi */
    printf("Unesi broj indeksa : ");
    scanf("%s",indeks);
    while (strcmp(indeks,"kraj") != 0)
    {
        printf("%s ",indeks);
        printf("Broj poena : %d\n",binary_search_stdlib(indeks));
        printf("Unesi broj indeksa : ");
        scanf("%s",indeks);
    }

    return 0;

}
```

Primer 35

5.4 Zadaci za vežbu:

Zadatak 15 *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 16 *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 17 *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).*

6

Programski jezik C

[1]

6.1 Alokacija memorije

Funkcija **void* malloc(unsigned n)** vraća pokazivač na blok od n susednih bajtova neinicijalizovane memorije ili `NULL` ukoliko zahtev ne može da se ispuni.

Funkcija **void* calloc(unsigned n, unsigned velicina)** vraća pokazivač na memorijski prostor dovoljno velik za n objekata navedene veličine, ili NULL ako zahtev ne može biti ispunjen. Rezervisani memorijski prostor se inicijalizuje nulama.

Funkcija **void* realloc(void *p, unsigned velicina)** menja veličinu objekta na koga pokazuje p tako što ga postavlja na vrednost velicina. Njegov sadržaj će ostati nepromenjen do manje od dve veličine – stare ili nove. Ako je nova veličina veća, novi prostor se ne inicijalizuje. Funkcija realloc vraća pokazivač na novi prostor, ili NULL ako zahtev ne može biti ispunjen i u tom slučaju *p ostaje nepromenjeno.

Funkcija **void free(void *p)** oslobađa memorijski prostor na koji pokazuje p. Na ne radi ništa ako je p jednako NULL. p mora biti pokazivač na memorijski prostor koji je prethodno alociran funkcijama calloc, malloc ili realloc.

Za korišćenje ovih funkcija neophodno je uključiti zaglavlje `stdlib.h`.

Primer 36 *Demonstracija funkcije* `malloc`

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

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

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

/* Kao da je moglo da se uradi
    int a[n];
```

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

```
*/
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 koristimo 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 37 *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);
}
```

6.2 Dinamički niz

Primer 38 *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 bismo 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 bismo 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;
    niz_reci=realloc(niz_reci, (alocirano)*sizeof(rec));

    /* Ovo je ekvivalentno kao da smo napisali sledeci blok naredbi:
    {
    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;
    }*/

    /* Nastavljamo dalje: */

    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); }
```

6.3 Niz pokazivača

Primer 39

```
#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 40

```
#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]);
}
```

6.4 Matrice

Primer 41 *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 42 *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);
}
```

Primer 43 *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.*

```
#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 44 *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);
}
```

6.5 Zadaci za vežbu

Zadatak 18 *(a) Napisati program koji omogućava unos dimenzije kvadratne matrice a zatim i unos elemenata te matrice sa standardnog ulaza.*

(b) Napisati funkciju koja računa zbir elemenata kvadratne matrice dimenzije $n \times n$.

(c) Napisati funkciju koja računa proizvod elemenata ispod glavne dijagonale matrice dimenzija $n \times n$.

(d) Napisati funkciju koja omogućava računanje proizvoda dve kvadratne matrice dimenzija $n \times n$.

(e) Napisati program koji omogućava unošenje dve kvadratne matrice i štampanje zbira, proizvoda elemenata te dve matrice kao i proizvoda elemenata ispod glavne dijagonale svake od tih matrica.

7

Programski jezik C

[1]

7.1 Liste

Primer 45 *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 46 *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 47 *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 48 *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 49 *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);
}
```

7.1.1 Dvosturko povezana kružna lista

Primer 50 *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);
}
```

7.2 Zadaci za vežbu

Zadatak 19 *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 20 ***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 21 *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.

8

Programski jezik C

[1]

8.1 Stek

Primer 51 *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");
}
```

8.2 Drveta

8.2.1 Binarno pretraživačko drvo

Primer 52 *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 53 *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 54 *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 55 *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 56 *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);

}
```

8.3 Zadaci za vežbu

Zadatak 22 ***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 23 ***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.*

9

Programski jezik C

[1]

9.1 Grafovi

NAPOMENA: Obavezno pogledati i primere (DFS numeracija, Acikličnost usmerenog grafa, DFS stablo, Topološko sortiranje , BFS numeracija, BFS stablo) sa sajta Milana Bankovića (`http://www.matf.bg.ac.yu/~milan/p2/dvocas_13/dvocas_13.pdf`)

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 57 *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)
```

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

```
{   int **matrica;
    int i;
    matrica=malloc(n*sizeof(int*));
    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 58 *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);
}
```

9.2 Zadaci za vežbu

Zadatak 24 *(**drugi kolokvijum 2006.**)Napisati program za:*

1. *formiranje orijentisanog grafa od n čvorova;*
2. *za par datih čvorova (čvorovi su zadati rednim brojevima) ispitati da li je jedan čvor dostupan iz drugog.*

NAPOMENA: Obavezno pogledati i primere (DFS numeracija, Acikličnost usmerenog grafa, DFS stablo, Topološko sortiranje , BFS numeracija, BFS stablo) sa sajta Milana Bankovića (`http://www.matf.bg.ac.yu/~milan/p2/dvocas_13/dvocas_13.pdf`)

10

Ispitni zadaci

Programiranje 1, april 2007. - grupa A

1. Data su dva celobrojna niza, a i b, uređena u rastućem poretku čije su dimenzije redom na i nb. Sastaviti funkciju koja formira niz c koji se sastoji od elemenata koji su zajednički nizovima a i b.

2. Sastaviti funkciju koja za zadatu nisku x formira nisku y koja se sastoji od alfabetskih karaktera niske x. Npr. za x="22.april 2007. god.", y="aprilgod".

3. Napisati funkciju koja proverava da li postoji element datog celobrojnog niza koji je jednak datom broju.

4. Sastaviti program koji sadržaj datoteke seminar.txt (koja sadrži neki ASCII-tekst) izlistava na ekranu i to u blokovima po 25 redova sa 80 karaktera u redu. Kad korisnik unese proizvoljni alfabetski znak sa tastature, ispisuje se sledeći ekran, i tako sve do kraja datoteke.

Programiranje 1, april 2007. - grupa B

1. Data su dva celobrojna niza, a i b, uređena u rastućem poretku čije su dimenzije redom na i nb i čiji su svi elementi različiti. Sastaviti funkciju koja formira niz c koji se sastoji od elemenata koji se javljaju u nizu a ali ne u nizu b.

2. Sastaviti funkciju koja za zadatu nisku x formira nisku y koja se sastoji od nealfabetskih karaktera niske x. Npr. za x="22.april 2007. god.", y="22. 2007.".

3. Napisati funkciju koja proverava da li su svi elementi datog celobrojnog niza jednaki datom broju.

4. Sastaviti program koji datoteku ulaz.txt (koja sadrži neki ASCII-tekst) prepisuje u datoteku izlaz.txt tako što zamenjuje svaki znak za kraj reda razmakom (blanko-simbolom). Ako se u rezultujućem tekstu pojavi više uzastopnih razmaka, zameniti ih jednim razmakom.

Osnovi programiranja, februar 2007. - grupa A

1. Reč je niska alfabetskih karaktera. Sastaviti program koji iz datoteke čita reči, a zatim ispisuje različite reči u obrnutom redosledu. Koristiti dinamičke strukture podataka u rešavanju.

2. Sastaviti funkciju koja za dati ceo broj tipa unsigned long int vraća

   (a) najmanji ceo broj koji se sastoji od istog broja 0 i 1 i koji je tipa unsigned long i

   (b) broj bitova postavljenih na 1.

3. Sastaviti program koji ispisuje prvih 100 elemenata Fibonačijevog niza zadatog sledećim formulama

$$f_1 = 1 f_2 = 2 f_n = f_{n-1} + f_{n-2} (n > 2)$$

   tačno u svakoj cifri. (Napomena: ako je f_n tipa unsigned long moguće je ispisati prva 43 elementa niza.)

4. Napisati funkciju

```
char * rstr_replace(char* str, char* substr, char* dstr);
```

   koja zamenjuje poslednje pojavljivanje date podniske (substr) u datoj niski (str) drugom niskom (dstr), i vraća početak druge niske u prvoj ili NULL ako se niska substr ne pojavljuje u niski str.

Programiranje 1, januar 2007. - grupa B

1. Dat je neuređen celobrojni niz **a** čija je dimenzija **na**. Sastaviti funkciju koja iz niza **a** isključuje sve ponovljene elemente i ažurira dimenziju niza.

2. Dve niske su anagrami ako se sastoje od istih karaktera. Sastaviti funkciju koja vraća 1 ako su niske x i y anagrami, inače 0. Npr. niske "vrata" i "trava" su anagrami.

3. Data je struktura

```
struct complex{
        float Re;
        float Im; }
```

   Ova struktura opisuje kompleksan broj čiji je realni deo Re a imaginarni Im. Sastaviti funkciju koja izračunava moduo proizvoda dva kompleksna broja. Sastaviti program koji testira ovu funkciju, a rezultat ispisuje u obliku "Moduo proizvoda (1. argument) i (2. argument) je (vrednost modula)".

4. Sastaviti program koji iz datoteke **ulaz.txt** (koja sadrži neki ASCII-tekst) prepisuje u datoteku **izlaz.txt** samo cifre i razmake (blanko), dok ostale karaktere zanemaruje. Po izvršenom kopiranju, program treba da štampa izveštaj o broju karaktera koji nisu prepisani.

5. Sastaviti program koji ispisuje prvih 10 elemenata Fibonačijevog niza zadatog sledećim formulama

$$f_1 = "a" f_2 = "ab" f_n = f_{n-1} * f_{n-2} (n > 2)$$

   gde * označava operator dopisivanja niski.

Programiranje 1, januar 2007. - grupa A

1. Data su dva celobrojna niza **a** i **b**, uređena u rastućem poretku čije su dimenzije redom **na** i **nb**. Sastaviti funkciju koja formira niz **c** koji se sastoji od elemenata nizova **a** i **b**, a koji je uređen takođe u rastućem poretku.

2. Sastaviti funkciju koja od dve zadate niske **x** i **y** formira treću nisku **z** koja se sastoji od zajedničkih karaktera niski **x** i **y**. Npr. x="Beograd" i y="Novi Sad", z="oad" ili z="ado" ili...(Formirati tačno jednu takvu nisku).

3. Data je struktura

```
struct tacka{
        float a;
        float b;
        char naziv[5];}
```

   Ova struktura opisuje tačku u ravni sa koordinatama **(a,b)** kojoj je dodeljeno ime **naziv**. Sastaviti funkciju **rastojanje** koja izračunava euklidsko rastojanje dve tačke tipa **struct tacka**. Sastaviti program koji testira funkciju **rastojanje**. Rezutat ispisuje u obliku "rastojanje od tacke (naziv 1) do tacke (naziv2) je (vrednost rastojanja)".

4. Sastaviti program koji iz datoteke **ulaz.txt** (koja sadrži neki ASCII-tekst) prepisuje u datoteku **izlaz.txt** samo alfabetske karaktere i razmak (blanko), dok ostale karaktere zanemaruje. Po izvršenom kopiranju, program treba da štampa izveštaj o broju prepisanih karaktera.

5. Sastaviti program koji ispisuje elemenate Fibonačijevog niza zadatog sledećim formulama

$$f_1 = 1 f_2 = 2 f_n = f_{n-1} + f_{n-2} (n > 2)$$

   Proceniti najveću vrednost za n ako se f deklariše kao **short int**.

Osnovi programiranja, januar 2007. - grupa A

1. Definišimo tipove Osoba i Skup na sledeći način:

```
struct osoba {
        char ime[MAXIME];
        unsigned int starost;}
struct skup {
        unsigned int dim;
        Osoba niz[MAX];}
typedef struct osoba Osoba;
typedef struct skup Skup;
```

   gde je MAXIME maksimalna dužina karakterske niske, a MAX maksimalan broj elemenata skupa. Pretpostavlja se da je niz u strukturi skup sortiran u rastućem redosledu (prvo po imenu, a zatim unutar istog imena po starosti). Sastaviti funkciju koja realizuje operaciju unije dva skupa i vraća uniju i 1 ili 0 prema tome da li je operacija uspešno obavljena ili ne.

2. Treba da obradimo datoteku **spisak** sa sledećom strukturom

   - na početku je broj n objekata koje sadrži datoteka, a za njim sledi
   - niz od n objekata sa strukturom Osoba (iz prethodnog zadatka)

Sastaviti funkciju koja čita i-tu osobu iz datoteke i vraća podatke o njoj .

3. U datoteci **program.c** nalazi se tekst programa na C-u. Sastaviti program koji formira listu svih deklarisanih promenljivih u programu **program.c** i ispisuje ih u leksikografskom poretku. (Promenljive su deklarisane isključivo deklaratorima char, int, float i u tekstu **program.c** nema ni pokazivača, ni funkcija). Koristiti dinamičke strukture podataka.

4. Sastaviti funkciju koja kao argumente uzima niske x i y i utvrđuje da li je x prefiks niske y. Ako jeste vraća kao povratnu vrednost 1, inače 0, a preko liste argumenata prefiks ("" ako ga nema).

5. Sastaviti program koji ispisuje elemenate Fibonačijevog niza zadatog sledećim formulama

$$f_1 = 1 f_2 = 2 f_n = f_{n-1} + f_{n-2} (n > 2)$$

Proceniti najveću vrednost za n ako se f deklariše kao **short int** .

Osnovi programiranja, januar 2007. - grupa B

1. Definišimo tipove Osoba i Skup na sledeći način:

```
struct osoba {
        char ime[MAXIME];
        unsigned int starost;}
struct skup {
        unsigned int dim;
        Osoba niz[MAX];}
typedef struct osoba Osoba;
typedef struct skup Skup;
```

gde je MAXIME maksimalna dužina karakterske niske, a MAX maksimalan broj elemenata skupa. Pretpostavlja se da je niz u strukturi skup sortiran u rastućem redosledu (prvo po imenu, a zatim unutar istog imena po starosti). Sastaviti funkciju koja ispituje da je jedan skup podskup drugog skupa i vraća 1 ako jeste, 0 inače.

2. Treba da obradimo datoteku **spisak** sa sledećom strukturom

   - na početku je broj n objekata koje sadrži datoteka, a za njim sledi
   - niz od n objekata sa strukturom Osoba (iz prethodnog zadatka)

   Sastaviti funkciju koja dodaje osobu na kraj datoteke.

3. U datoteci **program.c** nalazi se tekst programa na C-u. Sastaviti program koji formira listu svih izraza s desne strane operatora dodele u tekstu **program.c** i ispisuje ih u leksikografskom poretku. Pod izrazom se podrazumeva u ovom zadatku niska omeđena karakterima = i ; (U tekstu **program.c** nema ni pokazivača, ni funkcija). Koristiti dinamičke strukture podataka.

4. Sastaviti funkciju koja kao argumente uzima niske x i y i utvrđuje da li je x sufiks niske y. Ako jeste vraća kao povratnu vrednost 1, inače 0, a preko liste argumenata sufiks ("" ako ga nema)

5. Sastaviti program koji ispisuje prvih 10 elemenata Fibonačijevog niza zadatog sledećim formulama

$$f_1 = "a" f_2 = "ab" f_n = f_{n-1} * f_{n-2} (n > 2)$$

gde * označava operator dopisivanja niski.

Osnovi programiranja, oktobar 2006. - prva grupa

1. U ravni je zadato n krugova trojkama $\{x_i, y_i, r_i\}$, $i = 0, 1, \ldots, n-1$; pri tome su x_i, y_i koordinate centra, a r_i veličina poluprečnika kruga sa indeksom i.

   Napisati funkciju kojoj su argumenti dve trojke (koje određuju dva kruga), a koja vraća 1 (odnosno 0) ako se ova dva kruga seku (odnosno ako se ne seku); pretpostavlja se da krug obuhvata i kružnicu.

   Napisati program koji sa standardnog ulaza učitava broj $n \leq 100$ i trojke $\{x_i, y_i, r_i\}$, $i = 0, 1, \ldots, n-1$, a zatim ispisuje poruku da li se neka dva među krugovima (određenim ovim trojkama) seku.

2. *List* stabla je čvor koji nema ni jednog sina. Napisati funciju kojoj je argument pokazivač na koren binarnog stabla, a koja vraća broj listova u stablu.

3. Napistati program koji sa standardnog ulaza učitava brojeve m, n, a zatim u narednih m linija po n elemenata celobrojne matrice. Posle toga program treba da odštampa redni broj one vrste matrice koja ima najveći zbir elemenata; redni broj prve vrste je 1.

4. Napisati funkciju sa jednim celobrojnim argumentom n koja vraća 1 (odnosno 0) ako dekadne cifre broja n čine (odnosno ne čine) rastući niz.

5. *Palindrom* je niska koja glasi isto kad se čita spreda, kao i od pozadi — na primer niska "anavolimilovana". Napisati funkciju kojoj je argument niska `s`, a koja vraća dužinu najdužeg palindroma sadržanog u `s`. Na primer, za `s`="xyzanavolimilovanatuv" funkcija treba da vrati broj 15.

Osnovi programiranja, oktobar 2006. - druga grupa

1. U ravni je zadato n kvadrata (sa stranicama paralelnim osama) trojkama $\{x_i, y_i, a_i\}$, $i = 0, 1, \ldots, n-1$; pri tome su x_i, y_i koordinate centra, a a_i veličina stranice kvadrata sa indeksom i. Napisati funkciju kojoj su argumenti dve trojke (koje određuju dva kvadrata), a koja vraća 1 (odnosno 0) ako se ova dva kvadrata seku (odnosno ako se ne seku); pretpostavlja se da kvadrat obuhvata i granicu (stranice). Napisati program koji sa standardnog ulaza učitava broj $n \leq 100$ i trojke $\{x_i, y_i, a_i\}$, $i = 0, 1, \ldots, n-1$, a zatim ispisuje poruku da li se neka dva među kvadratima (određenim ovim trojkama) seku.

2. *Unutrašnji čvor* stabla je čvor koji ima bar jednog sina. Napisati funciju kojoj je argument pokazivač na koren binarnog stabla, a koja vraća broj unutrašnjih čvorova stabla.

3. Napistati program koji sa standardnog ulaza učitava brojeve m, n, a zatim u narednih m linija po n elemenata celobrojne matrice. Posle toga program treba da odštampa redni broj one kolone matrice koja ima najveći zbir elemenata; redni broj prve kolone je 1.

4. Napisati funkciju sa jednim celobrojnim argumentom n koja vraća 1 (odnosno 0) ako dekadne cifre broja n čine (odnosno ne čine) opadajući niz.

5. *Palindrom* je niska koja glasi isto kad se čita spreda, kao i od pozadi — na primer niska "anavolimilovana".

   Napisati funkciju kojoj je argument niska `s`, a koja vraća dužinu najdužeg palindroma sadržanog u `s`.

   Na primer, za `s`="xyzanavolimilovanatuv" funkcija treba da vrati broj 15.

Osnovi programiranja, septembar 2006. (grupa A)

1. Sastaviti program koji izračunava i ispisuje dužine rečenica datog teksta izražene brojem karaktera (simbol `.!?...` obeležava kraj rečenice ako se pojavi ispred velikog slova latinske abecede). Tekst se učitava iz datoteke ulaz.txt. Maksimalna dužina rečenice je 100.

2. Sastaviti program koji omogućava korisniku da unosi sa tastature niz celih pozitivnih ili negativnih brojeva kojih nema više od N. Za svaki uneti broj, negativne vrednosti se upisuju na uzastopne lokacije od početka niza, a pozitivne na lokacije počev od kraja niza. Unos se završava kada se unese 0 ili kada je niz popunjen. Kada je unos završen, program ispisuje sadržaj popunjenog dela niza.

3. Sastaviti funkciju koja iz date liste L izbacuje sva ponovljena pojavljivanja njenih elemenata. Npr. rezultat primene funkcije na listu `L=(b,a,b,c,a,c,a) je L=(b,a,c)`.

4. Neka se u datoteci f nalazi sadržaj nekog kataloga u datotečkom sistemu. Uz ime svake datoteke, naveden je i podatak o datumu poslednjeg pristupa datoteci kao jedan ceo broj (koji izražava broj proteklih sekundi od fiksiranog datuma). Sastaviti porgram koji čita podatke iz datoteke f i formira od njih uređeno binarno drvo. Svaki čvor u drvetu označava datoteku i sadrži podatke o imenu datoteke i datumu poslednjeg pristupa. Sastaviti funkciju koja obilazi drvo i ispisuje sve datoteke koje su nastale pre datuma zadatog kao argument funkcije.

5. Sa ulaza se učitava preusmeravanjem tekst T koji se sastoji od ascii-karaktera. Sastaviti funkciju koja pronalazi sve reči (=niske alfabetskih karaktera između separatora), a u kojima se pojavljuju bar jednom svi vokali (a, e, i, o, u).

Osnovi programiranja, jun 2006.

1. Visina nepraznog binarnog stabla jednaka je dužini najdužeg puta od korena do nekog lista: visina praznog stabla je -1. Napisati funkciju kojoj je argument pokazivač na koren binarnog stabla, a koja vraća visinu tog stabla, kao i program koji testira ovu funkciju.

2. Datoteka čije se ime učitava iz komandne linije sadrži spisak različitih reči nekog teksta (uređen prema ascii poretku), zajedno sa frekvencijom njihovog pojavljivanja. Napisati program koji čita sadržaj te datoteke i ispisuje na standardni izlaz frekvenciju pojavljivanja zadate reči (učitava se sa standardnog ulaza). Broj različitih reči nije vići od 1000. Za pretraživanje koristiti

   (a) bibliotečku funkciju `bsearch`

   (b) sopstvenu rekurzivnu funkciju binarnog pretraživanja

3. Napisati funkciju `int f(int n)` koja za dati prirodan broj `n` vraća zbir oktalnih cifara broja `n`.

4. Svaki red datoteke ULAZ.TXT sadrži ime, prezime i korisničko ime studenta na serveru ALAS (zabisano u formatu: dvoslovna oznaka smera, poslednje dve cifre godine upisa, četvorocifren broj indeksa, npr. mr050007, ml040950, aa030034). Napisati program koji generiše HTML datoteku generacija05.htm koja sadrži tabelu sa podacima o studentima upisanim 2005. godine.

5. Napisati program koji sa standardnog ulaza unosi pozitivan ceo broj `n` i ispisuje na standardni izlaz sumu bitova broja `n` na neparnim pozicijama. Smatrati da se bit najmanje težine nalazi na poziciji `0`.

Osnovi programiranja, drugi kolokvijum 2006.(I grupa)

1. Napisati funkciju za izračunavanje sume $\sum \frac{(-1)^n x^{2n} 2^{n+1}}{n! 3^n}$ za `n=1` do ∞ sa zadatom tačnošću ϵ. Iz standardne ulazne datoteke učitavaju se realni brojevi `x` i ϵ.

2. Napisati program za izračunavanje koeficijenata polinoma $S(x) = (x-a)P_n(x) + Q_m(x)$. Dati su: realan broj `a`, nenegativni celi brojevi `m` i `n` i koeficijenti polinoma $P_n(x)$ i $Q_m(x)$

3. Napisati program za:

   (a) formiranje orijentisanog grafa od n čvorova;

   (b) za par datih čvorova (čvorovi su zadati rednim brojevima) ispitati da li je jedan čvor dostupan iz drugog.

Osnovi programiranja, drugi kolokvijum 2006.(II grupa)

1. Napisati funkciju koja za date prirodne brojeve `k` i `n` vraća zbir `k`-tih stepena prvih `n` prirodnih brojeva.

2. Napisati program za izračunavanje koeficijenata polinoma $T_n(x)$ (`n` je dati prirodan broj) ako je poznato da važi $T_0(x) = 1$, $T_1(x) = x$, $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$.

3. Napisati program za:

   (a) formiranje orijentisanog grafa od n čvorova;

   (b) nalaženje svih čvorova datog grafa koji nisu dostupni iz zadatog čvora.

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

   $$c_i = \begin{cases} a_i, \quad akoa_i = b_i, \text{ a } a_i \text{ nije kontrolni znak – sa ASCII kodom} < 32 \\ \text{' ', ako } a_i = b_i, \text{ a } a_i \text{ jeste kontrolni znak'.', ako } a_i \neq b_i \end{cases}$$

   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 25 ***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 26 ***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 27 ***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 28 ***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 29 ***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 30 ***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 31 ***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 32 ***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 33 ***januar 2006.(II grupa)***

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

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

Zadatak 34 ***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 35 ***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 36 ***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 37 ***I kolokvijum, 18.januar 2006.(I grupa)*** *Neka je dat niz* `X` *od* `N` *nenegativnih celih brojeva. Sastaviti funkciju koja će iz niza* `X` *izbacivati sva pojavljivanja broja* `0` *i popunjavati ta mesta u nizu tako što će se preostali elementi niza pomerati ka početku niza. Odrediti i novu dimenziju* `N` *niza* `X`. *Npr. ulaz:* `N = 10, X = 0 22 11 2 0 17 33 4 0 999` → *izlaz :* `N = 7, X = 22 11 2 17 33 4 999`.

Zadatak 38 ***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 39 ***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 40 ***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 41 ***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 42 ***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 43 ***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 44 ***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 45 ***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 46 ***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 47 ***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 48 ***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 49 ***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 50 ***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 51 ***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 52 ***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 53 ***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 54 ***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 55 ***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 56 ***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 57 ***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 58 ***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 59 ***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 60 ***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 61 ***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 62 ***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 63 ***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 64 ***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 65 ***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 66 ***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 67 ***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 68 ***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 69 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 70 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 71 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 72 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 73 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 74 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 75 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 76 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 77 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 78 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 79 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 80 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 81 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 82 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 83 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 84 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 85 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 86 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 87 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 88 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 89 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 90 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 91 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 92 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 93 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 94 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 95 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.*