Programiranje 1

Beleške sa vežbi

Školska 2006/2007 godina

Matematički fakultet, Beograd

Jelena Tomašević

January 28, 2007

Sadržaj

1

Programski jezik C

[1]

1.1 Sortiranje niza

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

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

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

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

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

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

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

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

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

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

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

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

   putchar('\n');

   return 0;

}
```

1.2 Enumeracija

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

koriscenje:

int x=0;
boje b;

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

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

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

1.3 Strukture

Primer 2 *Napisati program koji izračunava obim i površinu trougla i kvadrata.*

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

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

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

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

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

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

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

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

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

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

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

    return o;
}

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

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

    return a;
}

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

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

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

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

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

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


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

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

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

}

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

Primer 3 *Ilustracija korisšćenja* `typedef`.

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

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

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

struct point
{
    int x;
    int y;
};

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

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

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

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

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

}

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

Primer 4 *Strukture se u funkcije prenose po vrednosti. Moguće je koristiti pokazivače na strukture.*

```
#include <stdio.h>

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

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

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

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


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

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

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

}
```

1.4 Rad sa datotekama

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

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

#include <stdio.h>

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

main()
{
        int i;
        int br;

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

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

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

        /* Zatvaramo datoteku */
        fclose(f);

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

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

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

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

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

        /* Zatvaramo datoteku */
        fclose(f);

}
```

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

   main()
   {
           FILE* datoteka;

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

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

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

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

   #define MAX_LINE 256

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

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

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

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

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

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

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

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

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

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

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

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

#include <stdio.h>

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

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

/* Maksimalni broj artikala */
#define MAX_ARTIKALA 1000

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

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

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

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

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

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

main()
{
        FILE* f;


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

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


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

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

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

Primer 5 *Program ilustruje čitanje etiketa iz neke HTML datoteke.*

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

/* Maksimalna duzina etikete */
#define MAX_TAG 100

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

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

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

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

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

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

    s[i]='\0';

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

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

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

 FILE* f;


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

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

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

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

fclose(f);
}
```

Primer 6 *Sa standardnog ulaza se učitava niz od* `n (n<100)` *tačaka u ravni takvih da nikoje tri tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka odredjuje konveksni mnogougao i rezultat ispisati na standardni izlaz.*

```
#include<stdio.h>

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

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

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

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

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

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

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

}
```

1.5 Formiranje HTML dokumenta

Primer 7 *Prilikm pokretanja programa koristiti redirekciju:*
`a.out >primer.html`
kako bi se rezultat rada programa upisao u datoteku `primer.html`.

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

Primer 8 *Napisati program koji generiše html dokument sa engleskim alfabetom.*

```
#include <stdio.h>
main()
{
```

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

Primer 9 *Napisati program koji generise html dokument koji prikazuje tablicu mnozenja za brojeve od 1 do 10.*

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

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

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

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

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

    printf("</tr>");
}
printf("</table>");
printf("</body></html>");
return 0;
}
```

1.6 Argumenti komandne linije

Primer 10 *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 11 *Program ispisuje opcije navedene u komandnoj liniji.* K&R *rešenje.*

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

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

#include <stdio.h>

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

}

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

Primer 12 *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 13 *Iz datoteke čije se ime zadaje kao argrument komandne linije, učitati cele brojeve sve dok se ne učita nula, i njihov zbir ispisati u datoteku čije se ime takođe zadaje kao argument komandne linije.*

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

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

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

Primer 14 *Datoteka cije se ime unosi sa komandne linije sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

```
#include <stdio.h>

#define MAXL 100
#define MAXN 1000

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

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

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

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

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

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

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

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

Zadaci za vežbu

Zadatak 1 *U datoteci brojevi.txt smeštena je prvo dimenzija niza a zatim i niz celih brojeva. Smatrati da nema više od 100 brojeva. Sa standardnog ulaza se učitava jedan ceo broj. Ispitati da li se taj broj nalazi u nizu brojeva učitanih iz datoteke brojevi.txt ili ne i rezultat ispisati na izlaz. Pri tome vršiti:*

a) linearnu

b) binarnu pretragu niza. Prethodno niz sortirati.

Zadatak 2 *Datoteka čije se ime unosi sa standardnog ulaza sadri podatke o uspehu studenata na kolokvijumima iz osnova programiranja. Prva linija datoteke sadrži broj studenata, a zatim svaka sledeća linija sadrži ime i prezime odredjenog studenta, njegov broj indeksa (u obliku korisničkog imena na alas-u npr. mr01123) i broj poena na prvom i na drugom kolokvijumu.*

a) Definisati strukturu podataka za čuvanje podataka o studentima

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

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

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

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

Zadatak 3 *Napisati program koji iz datoteke čije se ime unosi sa standardnog ulaza, učitava niz struktura tačaka, izračunava obim poligona određen učitanim nizom tačaka i ispisuje njegovu vrednost na standardni izlaz. Smatrati da u datoteci nema više od 100 tačaka.*