Osnovi programiranja

Beleške sa vežbi

Smer *Računarstvo i informatika*
Matematički fakultet, Beograd

Jelena Tomašević

March 28, 2006

Sadržaj

1

Programski jezik C

[1]

1.1 Strukture

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

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

Sintaksa strukture:

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

Primer 1 *Primer jednostavne strukture.*

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

Sada možemo deklarisati dve osobe na sledeći način:

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

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

```
struct licnost
{
```

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

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

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

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

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

Primer 2

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

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

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

Tada da bi pročitali starost neke ličnosti u nizu pišemo:

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

Može se definisati pokazivač na strukturu.

```
struct licnost *posoba;
```

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

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

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

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

Primer 3 *Elementi strukture mogu da budu i druge strukture.*

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

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

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

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

1.1.1 Operator typedef

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

```
typedef double RealanBroj;
```

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

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

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

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

Sada u kodu možemo da koristimo VelikiBroj kao tip.

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

```
typedef struct _licnost licnost;
```

Sada deklaracija može da bude:

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

Kao skraćen zapis za

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

typedef struct _tacka tacka;
```

može se koristiti:

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

Primer 4 *Struktura artikal.*

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

Primer 5 *Program ilustruje osnovne geometrijske algoritme kao i rad sa strukturama.*

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

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

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

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

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

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

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

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

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

```
}

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

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

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

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

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


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

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

   return povrsina;
}


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

```
    float o=0;

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

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

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

Primer 6 *Program koji učitava niz studenata i sortira ih po njihovim ocenama.*

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

#define MAX_IME  20

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

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

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

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

    word[i]='\0';
```

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

}

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

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

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

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

SelectionSort(studenti, br_studenata);

IspisiPodatke(studenti, br_studenata);
```

```
return 0;
}
```

Primer 7 *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");

}
```

1.2 Rad sa datotekama

Primer 8 *Program demonstrira otvaranje datoteka ("r" - read i "w" - write mod) i osnovne tehnike rada sa datotekama U datoteku se upisuje prvih 10 prirodnih brojeva, a zatim se iz iste datoteke citaju brojevi dok se ne stigne do kraja i ispisuju se na standardni izlaz.*

```
#include <stdio.h>

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

main()
{
        int i;
        int br;

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

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

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

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

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

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

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

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

        /* Zatvaramo datoteku */
        fclose(f);

}
```

Primer 9 *Program demonstrira "a" - append mod datoteka - nadovezivanje.*

```
#include <stdio.h>

main()
{
        FILE* datoteka;

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

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

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

Primer 10 *Program ilustruje rad sa datotekama. Program kopira datoteku ulaz.txt u datoteku izlaz.txt. Uz svaku liniju se zapisuje i njen broj.*

```
#include <stdio.h>

#define MAX_LINE 256
```

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

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

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

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

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

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

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

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

Primer 11 *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 12 *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);
}
```

1.3 Zadaci za vežbu

Zadatak 1 *Datoteka cije se ime unosi na ulazu sadrži podatke o studentima (ime, prezime, broj indeksa). Podaci su korektno zadati. Nema više od 1000 studenata. Prvo formirati niz struktura u memoriji, a onda ih ispisiati.*

Zadatak 2 *Definišemo strukturu VREME na sledeći način:*

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

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

Zadatak 3 ***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 na ulazu čita brojeve, gde su brojevi zapisani po jedan u svakom redu i u drugu datoteku čije se ime zadaje takođe na ulazu upisuje red po red date brojeve i njihovu multiplikativnu otpornost.*

Zadatak 4 ***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.*