Programiranje I

Beleške sa vežbi

Smer *Informatika*

Matematički fakultet, Beograd

Sana Stojanović

January 24, 2008

Sadržaj

1 Strukture

1. Napisati program koji korišćenjem struktura opisuje tačke u dvodimenzionalnom prostoru. Napisati i demonstrirati rad funkcija za izračunavanje rastojanja između dve tačke, obima i površine trougla.

```
#include <stdio.h>

/* Zbog funkcije sqrt. */
#include <math.h>
/* Napomena: 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 krajnjim tackama */
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
```

2. Ilustracija korišćenja *typedef* radi lakšeg rada sa strukturama.

```
#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 prilikom deklaracije 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
```

3. Napisati program koji sa standardnog ulaza učitava niz od n ($n < 100$) tačaka u ravni takvih da nikoje 3 tačke nisu kolinearne. Tačke se zadaju parom svojih koordinata (celi brojevi). Ispitati da li taj niz tačaka određuje konveksni mnogougao i rezultat ispisati na standradni 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");
}
```

2 Datoteke

1. Napisati program koji iz datoteke čije se ime zadaje kao prvi argument komandne linije učitava podatke o studentima (ime, prezime i broj indeksa). Pretpostavka je da su podaci u datoteci korektno zadati i da nema više od 1000 studenata. Prvo formirati niz struktura prilikom čitanja iz datoteke pa onda korišćenjem tog niza ispisati podatke o studentima na standardni izlaz.

```
#include <stdio.h>

#define MAXL 100
#define MAXN 1000

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

main(int argc, char *argv[])
{
    FILE *in;
    int j,i=0;
    student niz[MAXN];

    //Ako se broj argumenata komandne linije razlikuje od 2 onda znaci
    //da program nije dobro pozvan (ovaj deo ne morate koristiti na ispitu)
    if(argc!=2)
    {
     fprintf(stderr,"Neispravno pozivanje! Koriscenje: %s <ime datoteke>\n",
```

```
                argv[0]);
     return -1;
    }
    if((in=fopen(argv[1],"r")) == NULL)
    {
     fprintf(stderr,"Ne mogu da otvorim datoteku %s za citanje.",argv[1]);
     return -1;
    }

    while(1)
    {
     fscanf(in,"%s %s %d",&niz[i].ime, &niz[i].prezime, &niz[i].indeks);
     i++;
     if (feof(in))
        break;
    }

    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);
}
```

2. Napisati program koji kao argumente komandne linije prima ime datoteke i karakter i ispisuje na standardni izlaz broj pojavljivanja datog karaktera u datoj datoteci.

3. Napisati program koji iz datoteke čije se ime zadaje sa standardnog ulaza čita podatke o artiklima prodavnice.

   Svaki proizvod se odlikuje sledećim podacima :

   bar-kod - petocifreni pozitivan broj

   ime - niska karaktera

   cena - realan broj zaokružen na dve decimale

   pdv - stopa poreza - realan broj zaokružen na dve decimale

   Pretpostavljamo da su podaci u datoteci korektno zadati.

   Pretpostavljamo da se u prodavnici ne prodaje vise od 1000 različitih 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 bar kod */
        fscanf(f, "%d", &(a->bar_kod));

        /* Ukoliko smo dosli do kraja datoteke prijavljujemo neuspeh */
        if (feof(f))
                return 0;

        /* Citamo ime proizvoda */
        fscanf(f, "%s", a->ime);
        /* Citamo cenu */
        fscanf(f, "%f", &(a->cena));
        /* Citamo stopu poreza */
        fscanf(f, "%f", &(a->pdv));

        /* Prijavljujemo uspeh */
        return 1;
}

/* 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(int argc, char* argv[])
{
    FILE* f;

    /* Ukoliko nije navedeno ime kao argument komandne linije,
       trazimo od korisnika da ga unese */
    if (argc<2)
    {
       /* 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);
           return 1;
         }
    }
    /* Ime datoteke je prvi argument komandne linije */
    else
     {
        /* Otvaramo datoteku i proveravamo da li smo uspeli */
        if ( (f = fopen(argv[1], "r")) == NULL)
          {
            printf("Greska : datoteka %s ne moze biti otvorena\n",
                    argv[1]);
            return 1;
          }
```

```
    }

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

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

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

3 Životni vek i oblast važenja promenljivih

1. Ilustracija globalnih, lokalnih i statičkih promenljivih.

```
#include <stdio.h>

/* Globalna promenjiva */
int a = 0;

/* Uvecava se globalna promenjiva a */
void increase()
{
    a++;
    printf("increase::a = %d\n", a);
}

/* Umanjuje se lokalna promenjiva a. Globalna promenjiva
   zadrzava svoju vrednost. */
void decrease()
{
    /* Ovo a je nezavisna promenjiva u odnosu na globalno a */
    int a = 0;
    a--;
    printf("decrease::a = %d\n", a);
}

void nonstatic_var()
{
    /* Nestaticke promenjive ne cuvaju vrednosti kroz pozive funkcije */
    int s=0;
    s++;
    printf("nonstatic::s=%d\n",s);
```

```
}

void static_var()
{
    /* Staticke promenjive cuvaju vrednosti kroz pozive funkcije.
       Inicijalizacija se odvija samo u okviru prvog poziva. */
    static int s=0;
    s++;
    printf("static::s=%d\n",s);
}

main()
{
    /* Promenjive lokalne za funkciju main */
    int i;
    int x = 3;

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

    for (i = 0; i<3; i++)
    {
        /* Promenjiva u okviru bloka je nezavisna od spoljne promenjive.
           Ovde se koristi promenjiva x lokalna za blok petlje koja ima
           vrednost 5, dok originalno x i dalje ima vrednost 3*/
        int x = 5;
        printf("for::x = %d\n", x);
    }

    /* U ovom bloku x ima vrednost 3 */
    printf("main::x = %d\n", x);

    increase();
    decrease();

    /* Globalna promenjiva a */
    printf("main::a = %d\n", a);

    /* Demonstracija nestatickih promenjivih */
    for (i = 0; i<3; i++)
        nonstatic_var();

    /* Demonstracija statickih promenjivih */
    for (i = 0; i<3; i++)
        static_var();
}
```

```
Izlaz iz programa:
main::x = 3
for::x = 5
for::x = 5
for::x = 5
main::x = 3
increase::a = 1
decrease::a = -1
main::a = 1
nonstatic::s=1
nonstatic::s=1
nonstatic::s=1
static::s=1
static::s=2
static::s=3
```

4 Pokazivači

1. Uvođenje pojma pokazivača.

```
/* Pokazivaci - osnovni pojam */
#include <stdio.h>
main()
{
    int x = 3;

    /* Adresu promenjive x zapamticemo u novoj promeljivoj.
       Nova promenljiva je tipa pokazivaca na int (int*) */
    int* px;

    printf("Adresa promenljive x je : %p\n", &x);
    printf("Vrednost promenljive x je : %d\n", x);


    px = &x;
    printf("Vrednost promenljive px je (tj. px) : %p\n", px);
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Menjamo vrednost promenljive na koju ukazuje px */
    *px = 6;
    printf("Vrednost promenljive na koju ukazuje px (tj. *px) je : %d\n", *px);

    /* Posto px sadrzi adresu promenljive x, ona ukazuje na x tako da je
       posredno promenjena i vrednost promenljive x */
```

```
    printf("Vrednost promenljive x je : %d\n", x);


}
```

2. Napisati funkciju koja razmenjuje vrednosti svojim argumentima.

```
/* swap : Demonstracija prenosa argumenata preko pokazivaca */
#include <stdio.h>

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

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


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

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

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

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

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

```
    printf("Vrednost promenljive x je %d\n", x);
    printf("Adresa promenljive y je %p\n", &y);
    printf("Vrednost promenljive y je %d\n", y);

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

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

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

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

3. Ilustracija pokazivača na strukture.

```
/* Strukture se u funkcije prenose po vrednosti.
   Moguce je koristiti pokazivace 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);

}
```

5 Generisanje slučajnih brojeva

1. Napisati program koji generiše niz slučajnih realnih brojeva od 0 do 1.

```
#include<stdio.h>
#include<stdlib.h>
/* Zbog fje rand() */

/* Funkcija frand() vraca proizvoljan realan broj iz opsega 0-1.
   Koristimo funkciju rand() koja vraca ceo broj iz opsega 0-RAND_MAX */
float frand()
{
   return rand()/(RAND_MAX + 1.0);
}

/* Funkcija koja generise niz slucajnih realnih brojeva od 0-1 */
void frand_array(float a[], int n)
{
   int i;
```

```
   for(i=0; i<n; i++)
      a[i] = frand();
}

void print_array(float a[], int n)
{
   int i;
   for (i = 0; i < n; i++)
      printf("%f ",a[i]);
   putchar('\n');
}

main()
{
   float a[20];
   int n, i;

   printf("Koliko slucajnih brojeva (<20) zelite da generisete?  \n");
   scanf("%d", &n);

   frand_array(a, n);
   print_array(a, n);
}
```

2. Napisati program koji generiše niz slučajnih celih brojeva iz opsega $[1, 10]$.

```
/* Funkcija koja generise niz slucajnih brojeva iz opsega [1,10] */

void rand_array(float a[], int n) {
   int i;

   for(i=0; i<n; i++)
      a[i] = 1 + (int)(10.0 * frand());
}
```