Programiranje 1

Beleške sa vežbi

Školska 2008/2009 godina

Matematički fakultet, Beograd

Jelena Graovac (Tomašević)

January 15, 2009

Sadržaj

1

Programski jezik C

[1]

1.1 Pokazivač na funkciju

Primer 1 *Napisati funkciju koja izračunava sumu kvadrata, kubova i parnih brojeva na interbalu [a,b] sa korakom k.*

```
#include <stdio.h>
int kvadrat(int n) { return n*n; }
int kub(int n) { return n*n*n; }
int parni_broj(int n) { if (n%2==0) return n; else return 0; }

/* Funkcija izracunava sumu od 1 do n f(i), gde je f data funkcija.*/

int sumiraj(int (*f) (int), int a, int b, int k) {

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

int i, suma=0;
for (i=a; i<=b; i+=k)
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 2 do 8 sa korakom 3 je %d\n", sumiraj(kvadrat,2,8,3));
printf("Suma kubova brojeva od 3 do 13 sa korakom 2 je %d\n", sumiraj(kub,3,13,2));
printf("Suma parnih brojeva od 3 do 9 sa korakom 1 je %d\n", sumiraj(parni_broj,3,9,1));
}
/*
Izlaz:
Suma kvadrata brojeva od 2 do 8 sa korakom 3 je 93
Suma kubova brojeva od 3 do 13 sa korakom 2 je 4752
```

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

```
Suma parnih brojeva od 3 do 9 sa korakom 1 je 18
*/
```

1.2 Strukture

Struktura je kolekcija nekoliko promenljivih, potencijalno različitih tipova, koje su grupisane pod jednim imenom radi lakšeg rada sa njima.

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;
```

```
};

/* Moze se zapisati i
typedef struct point
{
    int x;
    int y;
} POINT;
*/

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);

}
```

Napomena: Ako je `tacka` ime promenljive tipa `POINT` (`POINT tacka;`) onda se `x` i `y` clanu pristupa kao `tacka.x` i `tacka.y`. Ako je `ptacka` ime promenljive tipa pokazivač na `POINT` (`POINT *ptacka;`) onda se `x` i `y` članu pristupa kao `ptacka->x` i `ptacka->y`.

Primer 5 *Napisati funkciju koja sabira dva kompleksna broja i rezultat vraća kao povratnu vrednost. Napisati funkciju koja oduzima dva kompleksna broja i rezultat vraća preko liste argumentata. Napisati program koji testira rad ovih funkcija.*

```
#include<stdio.h>
typedef struct complex
{
float Re,Im;
} COMPLEX;

COMPLEX saberi(COMPLEX prvi, COMPLEX drugi)
{
COMPLEX rezultat;
rezultat.Re = prvi.Re + drugi.Re;
rezultat.Im = prvi.Im + drugi.Im;
return rezultat;
}

void oduzmi(COMPLEX prvi, COMPLEX drugi, COMPLEX *rezultat)
```

```
{
rezultat->Re = prvi.Re - drugi.Re;
rezultat->Im = prvi.Im - drugi.Im;
}

main()
{
COMPLEX k1,k2,k3;
printf("Unesi realni i imaginarni deo prvog kompleksnog broja:\n");
scanf("%f %f", &k1.Re, &k1.Im);
printf("Unesi realni i imaginarni deo drugog kompleksnog broja:\n");
scanf("%f %f", &k2.Re, &k2.Im);
k3=saberi(k1,k2);
printf("(%.2f + %.2f *i) + (%.2f + %.2f *i) = (%.2f + %.2f *i) ",
                              k1.Re, k1.Im, k2.Re, k2.Im, k3.Re, k3.Im);
oduzmi(k1,k2,&k3);
printf("(%.2f + %.2f *i) - (%.2f + %.2f *i) = (%.2f + %.2f *i) ",
                              k1.Re, k1.Im, k2.Re, k2.Im, k3.Re, k3.Im);
}
```

Primer 6 *Uneti niz osoba, koje se karakterišu svojim imenom i brojem godina, sortirati po imenu a unutar istog imena, po starosti.*

```
#include<stdio.h>
#include<string.h> /*Zbog funkcija za rad sa stringovima*/
#define MAXIME 15

typedef struct osoba
{
char ime[MAXIME]; /*Mora se ograniciti duzina niza!*/
int starost;
} OSOBA;


main()
{
OSOBA nizOsoba[100];
OSOBA pom;
int n, i, j;
printf("Unesi broj osoba manji od 100\n");
scanf("%d", &n);
printf("Unesi %d osoba:\n", n);
for(i=0; i<n; i++)
         scanf("%s %d", nizOsoba[i].ime, &nizOsoba[i].starost);

for(i=0; i<n-1; i++)
         for(j=i+1; j<n; j++)
         /*Ukoliko je ime i-te osobe leksikografski ispred (vece od)
         imena druge osobe ili ako su istog imena i
         i-ta osoba je starija od j-te...*/
         if(strcmp(nizOsoba[i].ime, nizOsoba[j].ime) >0 ||
           (strcmp(nizOsoba[i].ime, nizOsoba[j].ime)==0 &&
            nizOsoba[i].starost > nizOsoba[j].starost))
```

```
        /*...razmeni mesta i-toj i j-toj osobi.*/
        {
          strcpy(pom.ime, nizOsoba[j].ime);
          pom.starost = nizOsoba[j].starost;
          strcpy(nizOsoba[j].ime, nizOsoba[i].ime);
          /* Ne bi smelo nizOsoba[j].ime=nizOsoba[i].ime; !!!*/
          nizOsoba[j].starost = nizOsoba[i].starost;
          strcpy(nizOsoba[i].ime, pom.ime);
          nizOsoba[i].starost = pom.starost;
        }
printf("Sortiran niz je:\n");
for(i=0; i<n; i++)
printf("%s %d\n", nizOsoba[i].ime, nizOsoba[i].starost);
}
```

Napomena: Ne bi smelo da se napise `nizOsoba[i].ime>nizOsoba[j].ime;` kada se porede dve niske već mora da poređenje obavi pozivom funkcije `strcmp`. Kod dodele se poziva funkcija `strcpy`!!!

1.3 Unije

Unija je struktura koja može čuvati (u različito vreme) objekte različitih tipova i veličina. Time se obezbeđuje manipulisanje različitim vrstama podataka u istom memorijskom području.

Primer 7 *Ilustracija koriscenja unije.*

```
#include <stdio.h>
typedef union u{
      int i;
      float f;
      char c;
      } u;

main(){
u unija;
unija.c='A';
unija.i=5;
/* Dozvoljen je pristup samo poslednje dodeljenom clanu unije */
printf("Trenutna vrednost unije je %d\n",unija.i); /* U redu je */
/* Pogresno bi bilo da se napise
printf("Trenutna vrednost unije je %c\n",unija.c);
*/
}
```

Unija se dakle može shvatiti kao struktura u kojoj svi članovi imaju relativnu poziciju 0 od njenog početka. Struktura je dovoljno velika kako bi mogla da čuva "najvećeg" člana.

Zadaci za vežbu:

Zadatak 1 *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. mr07888) 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. ***Napomena: Zadatak uraditi sa preusmeravanjem!***

Zadatak 2 *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. Zadatak uraditi sa preusmeravanjem!*

Zadaci za praktikum:

Zadatak 3 *Napisati program kojim se izdvajaju sekvence od n karaktera iz seminarskog rada. Broj n se učitava sa ulaza.*

Zadatak 4 *Napisati program kojim se izdvaja najduža sekvenca karaktera bez ponovljenih konsonanata.*

Zadatak 5 *Napisati funkciju koja datoteci brKljucRec.c vrši prebrojavanje pojave ključnih reči if, while i for i rezultat vraća preko liste argumenata. Napisati program koji vrši testiranje rada ove funkcije i sačuvati ga u datoteci sa imenom brKljucRec.c.*