

PROGRAMIRANJE 1

PROGRAMSKI JEZIK

C

Staša Vujičić
Čas 13

STRUKTURE - OSNOVNI POJMOVI

- Struktura je kolekcija nekoliko promenljivih, potencijalno različitih tipova, koje su grupisane pod jednim imenom radi lakšeg rada sa njima.
- Članovima strukture pristupa se operatorom "tačka" (npr a.x). Ovaj operator ima najviši mogući prioritet.

STRUKTURE - OSNOVNI POJMOVI

- ◉ Jedine operacije koje su dozvoljene nad objektom strukturnog tipa su:
 - dodela vrednosti strukturnog tipa promenljivoj tog tipa.
 - uzimanje adrese operatorom & (čime se dobija pokazivač na strukturu)
 - pristup članovima strukture, operatorom tačka.

ODNOS STRUKTURA I FUNKCIJA

- Kada je u pitanju odnos struktura i funkcija, važe sledeća pravila:
 - struktura se može prenositi kao argument funkcije. Pri tom se prenos vrši po vrednosti (kreira se lokalna kopija)
 - struktura se može vraćati kao povratna vrednost funkcije.

PRIMER 1

- ◉ *Napisati program koji izračunava obim i površinu trougla i kvadrata.*

```
/* Program uvodi strukture - geometrijske figure */
```

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

```
#include <math.h> /* Zbog funkcije sqrt */
```

```
/* 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 površinu 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;}
```

```
struct point srediste (struct point a, struct  
    point b)
```

```
{
```

```
    struct point s;
```

```
    s.x = (a.x + b.x) / 2;
```

```
    s.y = (a.y + b.y) / 2;
```

```
    return s;
```

```
}
```

```
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 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 moguće odrediti i putem sizeof */
printf("Povrsina kvadrata je %f\n",
area(square, sizeof(square)/sizeof(struct point)));
}
```

○ Izlaz:

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 2

- ◉ *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);
}
```

PRIMER 3

- ◉ *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 argumenata.*
- ◉ *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 4

- ◉ *Uneti niz osoba, koje se karakterišu svojim imenom, oznakom pola (m ili z) i brojem godina i potom:*
- ◉ *ispisati podatke za najstarijeg studenta a ukoliko ima više studenata sa maksimalnim brojem godina, ispisati podatke za onog studenta čije je ime prvo u leksikografskom poretku. Npr, ako su uneti sledeći podaci: Maja 18, Ana 19 i Petar 19, treba ispisati Ana 19, jer je niska Ana leksikografski ispred niske Petar*
- ◉ *Izračunati koliko u nizu ima ženskih osoba.*

PRIMER 5

- Krugovi i tačke.

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

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

```
/* Struktura tacka koja predstavlja tacku u  
euklidskoj ravni */
```

```
struct tacka {
```

```
    double x;
```

```
    double y;
```

```
};
```

```
/* Struktura krug sadrzi centar i poluprecnik  
kruga. Centar kruga je podatak tipa struct  
tacka. Ovim demonstriramo da clan strukture  
moze biti podatak bilo kog tipa, pa i druga  
struktura */
```

```
struct krug {  
    struct tacka o;  
    double r;  
};
```

```
/* Funkcija racuna rastojanje izmedju dve tacke.  
Ova funkcija demonstrira prenos struktura preko  
pokazivaca. Ovaj pristup treba uvek koristiti, cak  
i kada nemamo nameru da menjamo vrednost  
podatka unutar funkcije, zato sto je prenos  
pokazivaca brzi nego kopiranje citavog objekta  
strukturnog tipa. */
```

```
double rastojanje(struct tacka *a, struct tacka *b)
```

```
{
```

```
/* Obratiti paznju na pristup clanovima strukture  
preko pokazivaca i operatora strelica */
```

```
return sqrt((a->x - b->x) * (a->x - b->x) + (a->y -  
b->y) * (a->y - b->y));
```

```
}
```

```
/* Funkcija utvrđuje da li tacka p pripada
   krugu k. */
int pripada_krugu(struct tacka *p, struct krug
   *k)
{
    if(rastojanje(p, &k->o) == k->r)
        return 1;
    else
        return 0;
}
```

```
/* Funkcija main */  
int main ()  
{  
  
    struct tacka p;  
    struct krug k;  
  
    /* Ucitavamo koordinate tacaka i poluprecnik  
    kruga */  
  
    printf("Uneti koordinate tacke p: ");  
    scanf("%lf%lf", &p.x, &p.y);
```

```
printf("Uneti koordinate centra kruga: ");  
scanf("%lf%lf", &k.o.x, &k.o.y);
```

```
printf("Uneti poluprecnik kruga: ");  
scanf("%lf", &k.r);
```

```
/* Ispitujemo da li tacka pripada krugu i  
ispisujemo odgovarajuće poruke. */  
if(pripada_krugu(&p, &k))  
    printf("Tacka pripada krugu\n");  
else  
    printf("Tacka ne pripada krugu\n");  
}
```

UNIJE - OSNOVNI POJMOVI

- ◉ 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.
- ◉ Unija za razliku od stukture zauzima samo onoliko prostora koliko je dovoljno za smeštanje njenog najvećeg člana. Zbog toga je u svakom trenutku moguće koristiti samo jedan od članova unije.

PRIMER 1

```
#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);
```

```
*/
```

```
}
```

PRIMER 2

- ⦿ /* Ova unija moze da sadrzi ceo ili realan broj (u jednom trenutku samo jedno od ta dva) */

```
typedef union ceo_ili_realan {  
    int ceo;  
    double realan;  
} Ceo_ili_realan;
```

```
/* Funkcija main */
int main ()
{
    Ceo_ili_realan x;

    /* Koristimo x kao ceo broj */
    printf("Uneti ceo broj: ");
    scanf("%d", &x.ceo);
    printf("Celobrojna vrednost: %d\n", x.ceo);

    /* Koristimo x kao realan broj */
    printf("Uneti realan broj: ");
    scanf("%lf", &x.realan);
    printf("Realna vrednost: %f\n", x.realan);
}
```