Osnovi programiranja

2023/24

Strukture

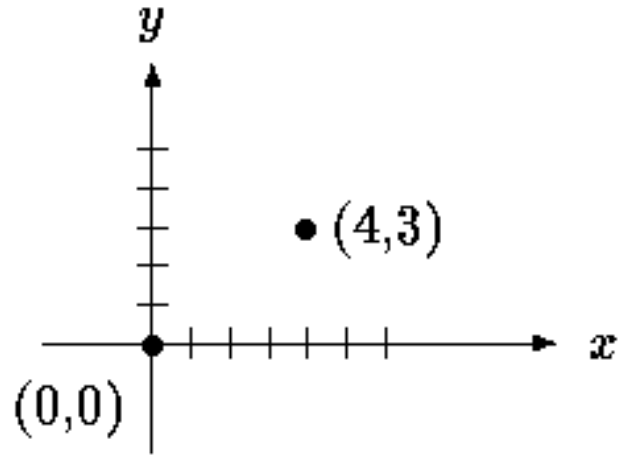
Strukture (slogovi)

- Ovakva organizacija podataka u većini programskih jezika omogućena je korišćenjem slogovnog tipa podataka. **Slog** je skup više promenljivih koje su grupisane pod zajedničkim imenom radi lakše manipulacije.
- Slogovi pomažu u organizovanju složenih podataka tako što omogućavaju grupisanje međusobno povezanih promenljivih na takav način da se one posmatraju kao celina, a ne kao pojedinačni entiteti. Promenljive unutar sloga nazivamo **polja** ili **promenljive članice**.
- Slogovi se u programskom jeziku C češće nazivaju strukturama i mogu se definisati korišćenjem rezervisane reči **struct** iza koje sledi naziv strukture i rezervisan znak **{**. Nakon toga ide spisak promenljivih članica. Definicija strukture se završava rezervisanim znakom **}**, nakon čega sledi znak **;**.
- Moguće je strukturu definisati i kao tip korišćenjem rezervisane reči **typedef**. Razlika je u tome što se naziv tipa zadaje posle rezervisanog znaka **}**, nakon čega sledi znak **;**.

Strukture

```
struct <naziv_strukture>
{
   <tip_polja_1> <polje_1>;
   <tip_polja_2> <polje_2>;
    ...
   <tip_polja_n> <polje_n>;
};
typedef struct
{
   <tip_polja_1> <polje_1>;
   <tip_polja_2> <polje_2>;
    ...
   <tip_polja_n> <polje_n>;
} <naziv_tipa>;
```

Strukture




```
struct point
{
  int x;
  int y;
};
```

Strukture se deklarišu pomoću ključne reči `struct`

Naziv strukture može se pisati iza reči `struct`

Promenljive članice strukture se navoda između vitičastih zagrada

Promenljive članice mogu imati isti naziv kao i obične promenljive ili kao članice neke druge strukture

Strukture

- Deklaracija `struct` definiše tip

Iza deklaracije strukture može se nalaziti lista promenljivih tog tipa

```
struct { ... } x, y, z;
```

potpuno analogno kao

```
int x, y, z;
```

- Deklaracije strukture iza koje se ne nalazi lista promenljivih ne izaziva rezervisanje memorije. Ona predstavlja šablon strukture.

Naziv neke strukture se može iskoristiti za deklarisanje primeraka te strukture

Ako je prethodno deklarisana struktura `point`, onda

```
struct point pt;
```

deklariše promenljivu `pt` koja je struktura tipa struct `point`

Struktura se može inicijalizovati navođenjem inicijalnih vrednosti

```
struct point maxpt = { 320, 200 };
```

Strukture

- Član određene strukture se može upotrebiti korišćenjem oblika

```
ime_strukture.clan
```

- Na primer, koordinate tačke pt možemo prikazati pomoću

```
printf("%d,%d", pt.x, pt.y);
```

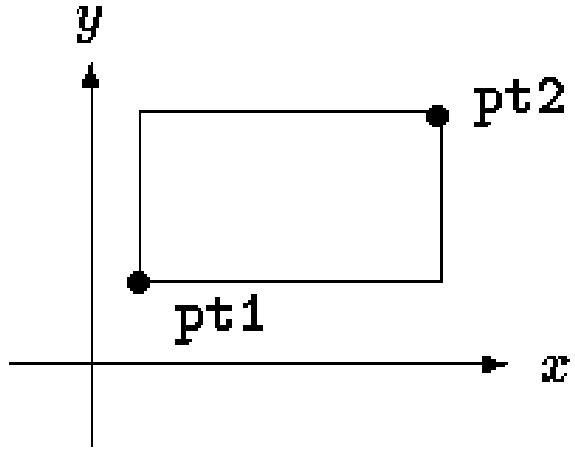
- Slično, rastojanje od koordinatnog početka do tačke pt možemo izračunati kao

```
dist = sqrt((double)pt.x * pt.x + (double)pt.y * pt.y);
```

Strukture

- Članice struktura mogu biti takođe strukture

```
struct rect
{
  struct point pt1;
  struct point pt2;
};
```



- Sada možemo deklarisati promenljivu screen

```
struct rect screen;
```

pa bi onda koordinata x prve tačke pravougaonika bila

```
screen.pt1.x;
```

Primer 1

- Napisati program koji izracunava obim i površinu trogula i kvadrata u koordinatnoj ravni.

```
#include <stdio.h>
#include <math.h>
struct point {
   int x;
   int y;  };
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);
}
float Heron(struct point A, struct point B, struct point C) {
   float a = segment_length(B, C);
   float b = segment_length(A, C);
   float c = segment_length(A, B);
   float s = (a+b+c)/2;
   return sqrt(s*(s-a)*(s-b)*(s-c));
}
```

Primer 1

```
float circumference(struct point polygon[], int num) {
   int i;
   float o = 0.0;
   for (i = 0; i<num-1; i++)
      o += segment_length(polygon[i], polygon[i+1]);
   o += segment_length(polygon[num-1], polygon[0]);
   return o;
}

float area(struct point polygon[], int num) {
   float a = 0.0;
   int i;
   for (i = 1; i < num -1; i++)
      a += Heron(polygon[0], polygon[i], polygon[i+1]);
   return a;
}
```

```
main() {
   struct point a, b = {1, 2}, triangle[3];
   struct point square[4] = {{0, 0}, {0, 1}, {1, 1}, {1, 0}};
   a.x = 0; a.y = 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;

   printf("sizeof(struct point) = %ld\n", sizeof(struct point));

   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("Pov. trougla: %f\n", Heron(triangle[0], triangle[1], triangle[2]));
   printf("Pov. kvadrata: %f\n", area(square, sizeof(square)/sizeof(struct
point)));
}
```

Primer 2

```
enum status_studenta {aktivan, neaktivan};

struct student
{
    char ime[20];
    char adresa[30];
    char indeks[6];
    enum status_studenta status;
    float prosek;
};

struct student student1, student2;
```

```
enum status_studenta { aktivan, neaktivan};
struct adresa
{
    char ulica[20];
    int broj;
    char opstina[15];
};
struct ispit
{
    char predmet[20];
    int ocena;
};
struct student
{
    char ime[20];
    struct adresa prebivaliste;
    char indeks[6];
    enum status_studenta status;
    float prosek;
    struct ispit ispiti[100];
};
```

```
student1.prosek = 9.5;
student1.ispiti[2].ocena = 9;
student1.prebivaliste.broj = 62;
scanf("%s%f", student1.ime, &student1.prosek);
student2.prosek = student1.prosek;
printf("%4.2f\n", student2.prosek);
```

Strukture

- Pored pristupa pojedinačnim poljima radi čitanja i dodele vrednosti, moguće je korišćenje i čitavih struktura u operaciji dodele. Tako se, na primer, može napisati:

```
student2=student1;
```

- čime se praktično vrednosti svih polja promenljive `student1` dodeljuju odgovarajućim poljima promenljive `student2`.

- Polja određenog tipa se mogu koristiti u svim izrazima tog tipa. Na primer, polja realnog tipa se mogu koristiti u realnim izrazima:

```
student1.prosek=student2.prosek+0.5;
```

- Kao što smo mogli videti na primerima naredbi `scanf` i `printf`, strukture se mogu koristiti i kao parametri funkcija, na potpuno isti način kao i svi ostali tipovi podataka.

Primer 3

- Napisati program koji učitava podatke o fudbalskim ekipama, broj bodova koje su osvojili u toku sezone, kao i ukupan broj žutih kartona koji su dobili igrači svakog tima, a zatim na osnovu broja bodova štampa tabelu i određuje tim koji će biti nagrađen za "fer plej" (tim koji ima najmanje žutih kartona).

```
#include <stdio.h>
typedef struct
{
    char naziv[20];
    int bodovi;
    int kartoni;
} ekipa;

typedef ekipa niz_ekipa[100];
```

Primer 3

```
ekipa ucitaj_ekipu()
{
    ekipa e;
    printf("Unesite naziv ekipe:\n");
    scanf("%s", e.naziv);
    printf("Unesite broj ostvarenih bodova:\n");
    scanf("%d", &e.bodovi);
    printf("Unesite broj zutih kartona:\n");
    scanf("%d", &e.kartoni);

    return e;
}
```

Primer 3

```
void sortiraj(niz_ekipa lista, int n)
{
    int i, j;
    ekipa pom;

    for(i = 0; i < n-1; i++)
          for(j = i+1; j < n; j++)
          if(lista[i].bodovi < lista[j].bodovi)
          {
              pom = lista[i];
              lista[i] = lista[j];
              lista[j] = pom;
          }
}
```

Primer 3

```
int ferplej(niz_ekipa lista, int n)
{
    int i;
    int min_kartona, min_indeks;

    min_kartona = lista[0].kartoni;
    min_indeks = 0;

    for(i = 1; i < n; i++)
       if(lista[i].kartoni < min_kartona)
       {
          min_kartona = lista[i].kartoni;
          min_indeks = i;
       }
    return min_indeks;
}
```

Primer 3

```
void stampaj_tabelu(niz_ekipa lista, int n)
{
    int i;
    printf("Naziv Bodovi Kartoni\n");
    for(i = 0; i < n; i++)
    printf("%20s%10d%10d\n", lista[i].naziv,    lista[i].bodovi,
   lista[i].kartoni);
}
```

Primer 3

```
main()
{
 niz_ekipa lista;
 int n, i;

 printf("Unesite broj ekipa:\n");
 scanf("%d", &n);

 for(i = 0; i < n; i++)
          lista[i] = ucitaj_ekipu();

 sortiraj(lista, n);

 stampaj_tabelu(lista, n);

 i = ferplej(lista, n);

 printf("Ekupa nagradjena za fer plej je %s\n", lista[i].naziv);
}
```