

Programiranje I

Strukture

KORISNIČKI DEFINISANI TIPOVI PODATAKA

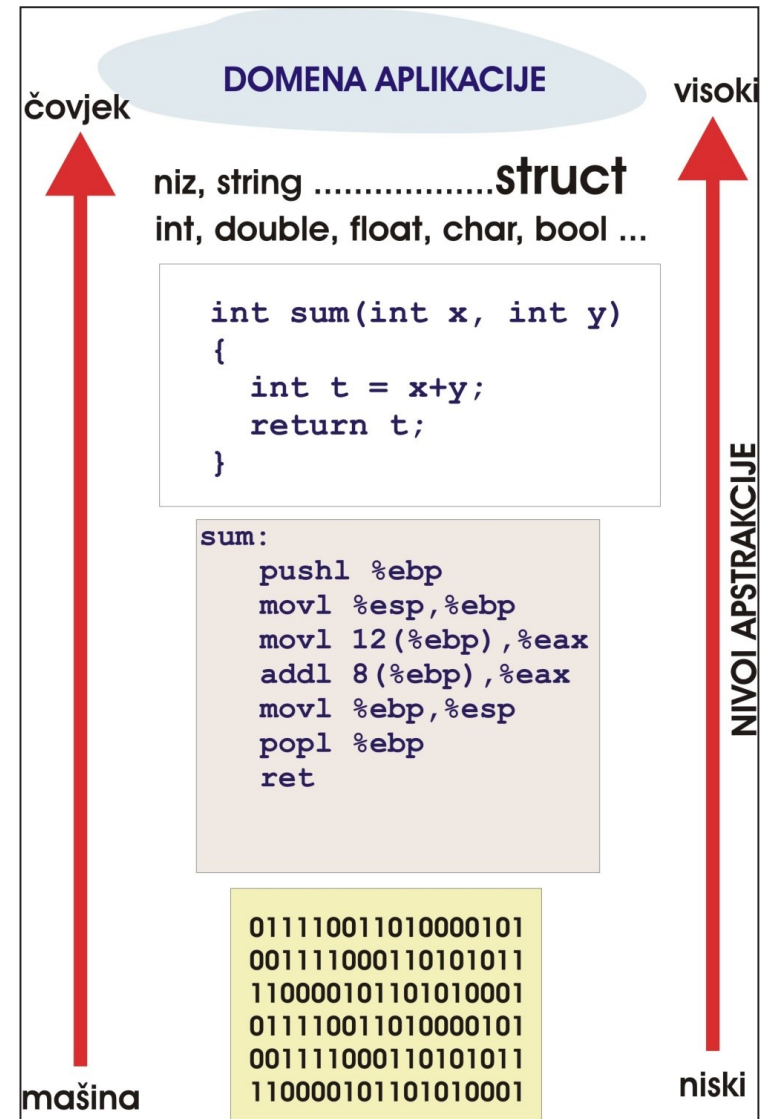


Strukturirani tipovi podataka

- Veoma često postoje “povezani” podaci različitih tipova koje je, zbog pogodnijeg pristupa pod jedinstvenim identifikatorom, potrebno zajedno pohraniti. Na taj način je moguće, u program koji razvijamo, direktno “mapirati” svojstva i ponašanje entiteta iz realnog svijeta.
- Mapiranje ili preslikavanje je omogućeno korištenjem “korisnički definisanih tipova podataka” koji se realizuju uz pomoć struktura (`struct`).
- Jedino čovjek posjeduje sposobnost apstrakcije – odabira bitnih aspekata nekog sistema i zanemarivanja onih koji su manje bitni.
- Posmatrajući nivoe apstrakcije kroz programiranje (razvoj softvera) u smjeru mašina – čovjek jasno je da idemo od niskog ka visokom nivou apstrakcije.

Nivoi apstrakcije

- Slika prikazuje nivoe apstrakcije krećući se od mašinskog, preko assembly jezika, pa sve do programskih jezika visokog nivoa, kakav je C++, koji je udaljeniji od mašinskog, a bliži čovjekovom jeziku.
- Posmatrajući sisteme iz realnog svijeta koje bi se trebali simulirati kroz softversku aplikaciju, moguće je uočiti da strukturu sistema sačinjavaju entiteti i odnosi među njima kao na primjer:
 - Bolnički sistem: *pacijent, ljekar, medicinski nalaz, bolesnički karton* itd.
 - Trgovačka radnja: *proizvod, dobavljač, kupac, proizvođač, ponuda, faktura* itd.
- Ideja je u tome da se entiteti iz okruženja preslikaju u programski kod što je omogućeno korištenjem korisnički definisanih tipova podataka.
- Korisnički definisani tipovi podataka su obično sastavljeni od prostih, ugrađenih tipova podataka.





Domena aplikacije

- Entitete iz stvarnog svijeta preslikavamo u programski kod i na taj način, uz pomoć ključne riječi `struct`, kreiramo korisnički definisane tipove podataka



```
struct Zaposlenik{  
    char JMBG[14];  
    char ime[25];  
    char prezime[25];  
    int godinaRodjenja;  
    //...  
};
```



```
struct Proizvod{  
    char sifra[30];  
    char naziv[30];  
    char proizvodjac [30];  
    int godinaProizvodnje;  
    float cijena;  
    //...  
};
```



```
struct Faktura{  
    char broj[30];  
    Proizvod proizvodi[30];  
    //...  
    float iznos;  
};
```



Struktura - korisnički definisani tip podatka

- Kao što nizovi grupišu veći broj podataka istog tipa, strukture služe za grupisanju više podataka obično različitih tipova
- Strukture omogućavaju kreiranje novih, korisnički definisanih tipova podataka, pa u narednom primjeru `Proizvod` postaje novi tip podatka
- Kreiranjem strukture se ne alocira memorijski prostor
- Memorijski prostor se alocira tek nakon kreiranja varijabli (objekata) koji su tipa te strukture

```
struct Naziv_Struktura{
    Tip_1 atribut_1;
    Tip_2 atribut_2;
    //....
    Tip_n atribut_n;

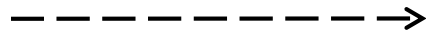
    tip funkcija_1(tip parametar);
    //....
};
```

```
struct Proizvod{
    char sifra[30];
    char naziv[30];
    char proizvođač [30];
    int godinaProizvodnje;
    float cijena;
    void setNaziv(char naziv[]);
    float getCijenu();
};
```



Korisnički definisani tip podatka

- Tip podatka – skup vrijednosti nad kojima se mogu primijeniti određene operacije i koji za pohranu zahtijeva određenu količinu memorije
- Ukoliko prethodnu definiciju preformulišemo i kažemo da vrijednosti predstavljaju svojstva (atribute) entiteta (objekta), a operacije (funkcije) odražavaju ponašanje entiteta, onda imamo sve što je potrebno da bi taj entitet “uveli” u računarski program



```
struct Proizvod{  
    //osobine entiteta predstavljaju  
    //članovi podaci - atributi  
    char sifra[30];  
    char naziv[30];  
    char proizvođač [30];  
    int godinaProizvodnje;  
    float cijena;  
    //ponašanje entiteta je definisano  
    //funkcijama članicama  
    void setNaziv(char naziv[]);  
    float getCijenu();  
};
```

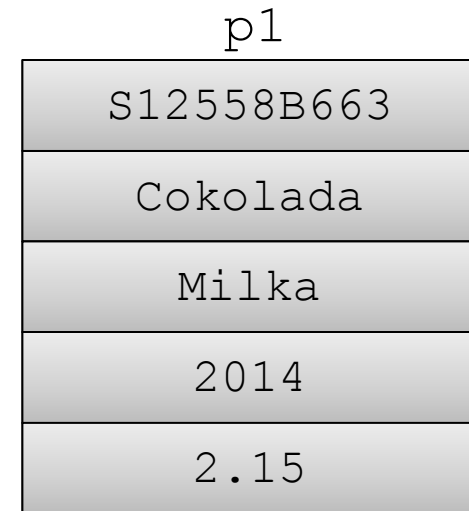


Korisnički definisani tip podatka

- Članovi podaci (atributi) korisnički definisanog tipa određuju strukturu objekta u memoriji

```
struct Proizvod{
    char sifra[30];
    char naziv[30];
    char proizvođač [30];
    int godinaProizvodnje;
    float cijena;
    void setNaziv(char naziv[]);
    float getCijenu();
};

void main(){
    //p1 je objekat tipa Proizvod
    Proizvod p1;.....
    //inicijalizacija atributa za p1...
}
```





Korisnički definisani tip podatka

- Odnos između strukture i njenih varijabli (objekata) se može predstaviti primjerom pečata (koji predstavlja strukturu) i njegovih otisaka (koji predstavljaju objekte).
- Svi objekti određen strukture (pečata) bi trebali imati ista obilježja ili attribute



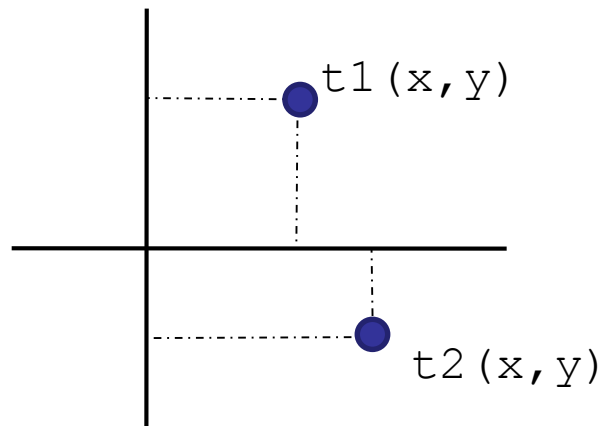
Struktura Tacka



//objekte je moguće kreirati i
//na kraju deklaracije strukture

```
struct Tacka{  
    float x;  
    float y;  
}t1,t2;
```

```
void main(){  
    t1.x = 1.2;  
    t1.y = 1.2;  
    t2.x = 1.5;  
    t2.y = -0.8;  
}
```



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

```
void main(){  
    Tacka t1;  
    Tacka t2;  
    t1.x = 1.2;  
    t1.y = 1.2;  
    t2.x = 1.5;  
    t2.y = -0.8;  
}
```



Inicijalizacija objekata strukture

- Vrijednosti atributa nekog objekta moguće je inicijalizovati na dva načina:
 - Neposredno prilikom deklaracije objekta

```
Proizvod p1={"S12558B663", "Cokolada", "Milka", 2014, 2.15};
```

- Inicijalizacijom svakog pojedinog atributa kojima se pristupa koristeći operator tačku (.)

```
Proizvod p1;  
p1.godinaProizvodnje = 2014;  
p1.cijena = 2.15;  
//funkcija strcpy vrsi kopiranje niza karaktera  
//p1.sifra = "S12558B663"; //←GRESKA  
strcpy_s(p1.sifra, "S12558B663");  
strcpy_s(p1.naziv, "Cokolada");  
strcpy_s(p1.proizvodjac, "Milka");
```



Inicijalizacija objekata strukture

- Nekoliko napomena koje se tiču struktura
 - Operator tačka (.) odvaja ime varijable (objekta) i ime člana strukture (atributa)
 - Operator tačka (.) spada u najvišu prioritetnu grupu operatora i ima asocijativnost slijeva na desno:
 - `++objekat.atribtut` je ekvivalentno `++(objekat.atribtut)`
 - `&objekat.atribtut` je ekvivalentno `&(objekat.atribtut)`
 - Kada struktura sadrži niz kao član strukture, onda se elementima niza pristupa izrazom

`objekat.atribtut[lokacija]`

- Na primjer:

```
if (p1.naziv[0] == 'C')  
    //...
```



Operacije nad objektima

- Pridruživanje (=)
- Uzimanje adrese (&), primjena `sizeof` operatora
- Objekti mogu biti argument funkcije
- Funkcija može vratiti objekat korisnički definisanog tipa

```
void Ispis(Tacka t){  
    cout<<"t.x = "<<t.x<<"\nt.y = "<<t.y<<endl;  
    cout<<"sizeof(t) = "<<sizeof(t)<<endl;  
    cout<<"&t = "<<&t<<endl;  
}
```

```
Tacka Suma(Tacka t1,Tacka t2){  
    float x = t1.x + t2.x;  
    float y = t1.y + t2.y;  
    Tacka temp = {x,y};  
    return temp;  
}
```

```
void main(){  
    Tacka t1 = {2.3, 1.8};  
    Tacka t2;  
    t2 = t1;  
    Tacka t3 = Suma(t1,t2);  
    Ispis(t3);  
}
```

Članovi strukture



- Pored ugrađenih tipova podataka, članovi strukture mogu biti i korisnički definisani tipovi podataka

```
struct Datum{
    int dan;
    int mjesec;
    int godina;
};
struct Proizvod{
    char sifra[30];
    char naziv[30];
    char proizvođač [30];
    int godinaProizvodnje;
    float cijena;
    Datum upotrebljivoDO;
};
```

```
void main(){
    Proizvod p1={"S12558B663", "Cokolada", "Milka", 2014, 2.15, 22, 5, 2015};
}
```

p1

S12558B663		
Cokolada		
Milka		
2014		
2.15		
22	5	2015
dan	mjesec	godina



KRAJ PREZENTACIJE