

Datum: 11.03.2015.

Vježbe 3

Strukture

U ovom materijalu je detaljnije predstavljen pojam i način korištenja struktura koje omogućavaju kreiranje novih tipova podataka. Pored pomenutog, za strukture se veoma često kaže da predstavljaju kolekciju varijabli različitih tipova podataka.

Pod konstatacijom da strukture predstavljaju kolekciju varijabli različitog tipa se podrazumijeva da su strukture pogodne za čuvanje više različitih informacija o nekom entitetu npr. informacija o igraču nekog tima: *ime*, *prezime*, *visina*, *težina*, *plata*, *broj pogodaka*, *broj asistencija* itd. Na prvi pogled je jasno da se pobrojane informacije ne mogu smjestiti u jedan niz, a razlog tome je činjenica da nizovi mogu sadržavati više elemenata koji moraju biti istog tipa. Idealan način za rješavanje problema koji zahtijevaju čuvanje veće količine podataka različitog tipa predstavlja upravo korištenje struktura.

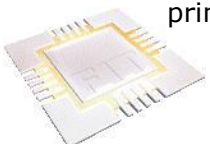
Pored ugrađenih tipova podataka (*int*, *float*, *double* itd.) strukture predstavljaju **korisnički-definisane tipove podataka**. Pri deklaraciji varijable nekog od ugrađenih tipova prvo se navodi tip podatka, pa tek onda ime varijable:

```
//tipPodatka  imeVarijable;  
int          mojaVarijabla;
```

U prethodnom primjeru je deklarirana varijablu koja se naziva *mojaVarijabla*, a u koju se može pohraniti neka cjelobrojna vrijednost. Slično je i sa strukturama, ali ipak postoje određene razlike. Razlike se najbolje mogu uočiti na nekom primjeru pa će na početku biti kreirana struktura čija „*varijabla*“ (za sada je nazivamo varijabla) može čuvati informacije o igraču nekog tima.

```
struct igračTima{  
    char ime[30];  
    char prezime[30];  
    float visina;  
    float težina;  
    float plata;  
    int brojPogodaka;  
    int brojAsistencija;  
};
```

Na osnovu prethodnog primjera se može zaključiti da se prilikom deklarisanja strukture koristi ključna riječ *struct*, nakon čega slijedi naziv strukture (u ovom slučaju *igračTima*). Naziv strukture se može posmatrati kao novi tip podatka. Također, bitno je primijetiti da se tijelo strukture nalazi unutar vitičastih zagrada koje završavaju znakom



tačka-zarez (;). Unutar tijela strukture se nalaze njeni elementi koji se nazivaju **obilježja** ili **atributi**. Struktura `igracTima` posjeduje sljedeća obilježja (navesti ćemo samo neka od njih):

- ✓ `char ime[30]` – niz u koji se može pohraniti 30 karaktera (ime igrača)
- ✓ `float visina` – varijabla u koju se pohranjuje visina igrača
- ✓ `int brojPogodaka` – varijabla u koju se pohranjuje broj pogodaka svakog igrača (bilo da se radi o golovima, koševima....)

Prilikom deklaracije obilježja strukture `igracTima` korištene su različite vrste ugrađenih tipova podataka. Kao tipovi podataka obilježja mogu se koristiti i drugi korisnički definisani tipovi, ali o tome će biti više riječi nešto kasnije. Dakle, može se zaključiti da strukture pomažu pri manipulaciji većom količinom podataka različitog tipa. Kreiranjem neke strukture ostvaruje se mogućnost kreiranja „varijabli“ koje su tipa te strukture. Varijable koje umjesto ugrađenih (`int`, `float`) koriste korisnički definisane tipove podataka (`igracTima`) najčešće nazivamo **objektima** ili **instancama** tih strukture. Da bi ovo bilo malo jasnije, u nastavku je kreiran objekat `Igrac1` koji je tipa strukture `igracTima`:

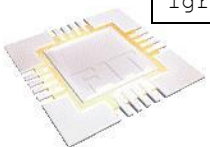
```
//nazivStrukture imeObjekta;
igracTima Igrac1;
```

Objekti određene strukture se deklariraju na isti način kao i varijable sa ugrađenim tipovima podataka. Znači, prvo se navodi naziv strukture (tip podatka), pa onda naziv objekta (varijable). U prethodnom dijelu teksta je kreirana struktura koja se naziva `igracTima` što je omogućilo kreiranje objekta `Igrac1`. Radi lakšeg razumijevanja, može se reći da je kreirana „varijabla“ `Igrac1` tipa `igracTima`.

```
#include <iostream>
using namespace std;
//struct kreira novi tip podatka koji se naziva igracTima
struct igracTima{
    //obilježja - atributi sturkture
    char ime[30];
    char prezime[30];
    float visina;
    float tezina;
    float plata;
    int brojPogodaka;
    int brojAsistencija;
}; //kompajler ce prijaviti gresku ukoliko izostavite tačku-zarez(;)
void main() {
    //kreiranje objekta tipa igracTima
    igracTima Igrac1;
    system("pause");
}
```

Prethodni program ne radi baš ništa korisno, ali se iz njega može vidjeti način na koji se kreira struktura (`igracTima`) i objekat (`Igrac1`) te strukture. Sada se postavlja pitanje na koji način inicijalizirati obilježja strukture, te kako ih koristiti? Za inicijalizaciju vrijednost obilježja nekog objekta prvo se navodi ime objekta, zatim znak tačku (.), te ime obilježja:

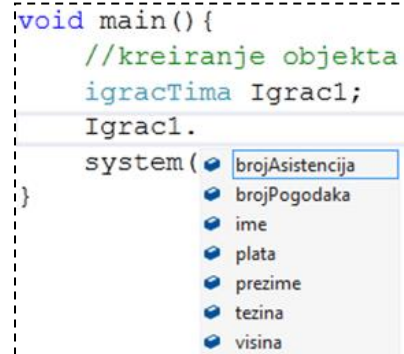
```
//imeObjekta.imeObiljezja = vrijednost;
Igrac1.plata = 670;
```



Prilikom inicijalizacije obilježja, većina razvojnih okruženja, kao što je Visual Studio, programerima pruža malu pomoć. Naime, odmah nakon navođenja imena objekta i znaka tačke, dobija se lista svih obilježja koji se nalaze u strukturi (što se vidi na sljedećoj slici).

Dakle, sva obilježja objekta se mogu inicijalizovati na sljedeći način (u nastavku je prikazana samo `main` funkcija):

```
void main() {
    //kreiranje objekta tipa igračTima
    igračTima Igracl;
    //inicijalizacija obilježja
    strcpy_s(Igracl.ime, "Refet");
    strcpy_s(Igracl.prezime, "Gojak");
    Igracl.visina = 1.98;
    Igracl.tezina = 99;
    Igracl.plata = 940;
    Igracl.brojPogodaka = 50;
    Igracl.brojAsistencija = 38;
    system("pause");
}
```



```
void main() {
    //kreiranje objekta
    igračTima Igracl;
    Igracl.
    system(
        brojAsistencija
        brojPogodaka
        ime
        plata
        prezime
        tezina
        visina
    )
}
```

Bitno je napomenuti da se obilježja tipa `char` (niz karaktera) inicijalizuju koristeći funkciju `strcpy_s()`. Prvi dio naziva funkcije **str** se odnosi na **string**, a drugi dio **cpy** predstavlja skraćenicu riječi **copy**. Dodatak `_s` označava da se radi o sigurnijoj verziji funkcije `strcpy()`. Pokušaj direktne inicijalizacije obilježja `ime` uzrokuje grešku prilikom kompajliranja:

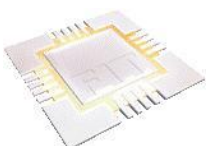
```
Igracl.ime = "Refet";
//Error! error C2440: '=': cannot convert from 'const char [6]' to 'char [30]'
```

Funkcija `strcpy_s()` će se puno detaljnije opisati u jednom od narednih materijala koji će obrađivati područje manipulisanja nizovima karaktera. Za sada je dovoljno znati da pomenuta funkcija prihvata dva argumenta; prvi kojim se definiše **gdje** se nešto kopira (destinaciju) i drugi koji definiše **šta** se kopira (izvor).

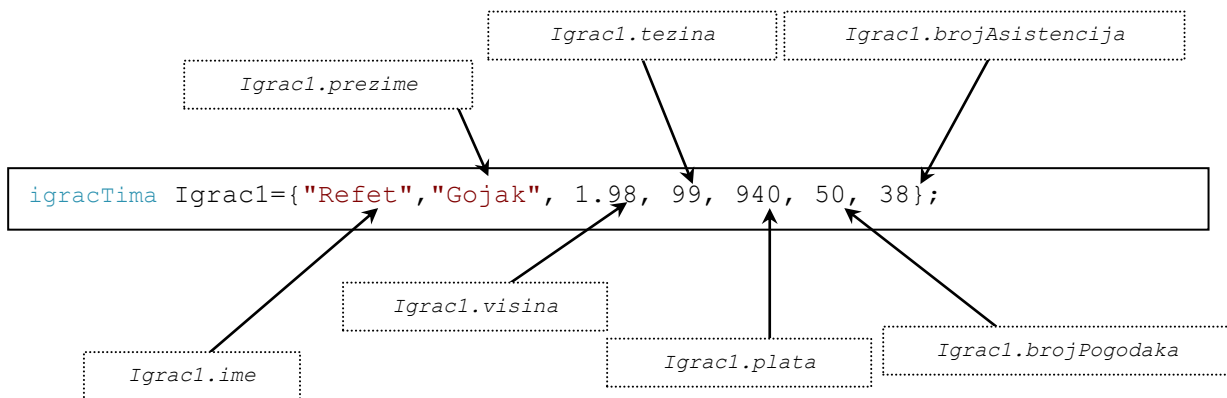
```
//      :destinacija:      :izvor:
strcpy_s(Igracl.ime, "Refet");
```

Nakon inicijalizacije vrijednosti svih njegovih atributa objekat `Igracl` se može predstaviti sljedećom slikom:

ime	Refet
prezime	Gojak
visina	1.98
tezina	99
plata	940
broj pogodaka	50
broj asistencija	38



Treba pomenuti činjenicu da se inicijalizaciju obilježja nekog objekta može izvršiti na više načina, a jedan od njih je da se odmah pri deklaraciji objekta, unutar vitičastih zagrada, navedu vrijednosti obilježja.



Nakon inicijalizacije, vrijednosti obilježja se mogu ispisati jednostavnim navođenjem imena objekta, znaka tačke (.), te imena obilježja (atributa).

```
//cout<<imeObjekta.imeObiljezja;
cout<<"Ime Igraca 1 je: "<<Igrac1.ime<<endl;
```

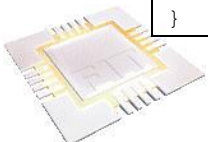
Naredni primjer obuhvata sve do sada objašnjeno, tako da bi vam programski kod trebao biti u potpunosti razumljiv. Zbog veličine programa, neke od obilježja se neće koristiti.

```
#include <iostream>
using namespace std;
char crt[]="\n-----\n";
struct igracTima{
    char imePrezime[40]; //ime i prezime ce uvijek biti odvojeno-dva atributa,
    //ali su u cilju reduciranja broja linija code-a smjesteni u jedan atribut
    float visina;
    float plata;
    int brojPogodaka;
};
void main(){
    //kreiramo prvi objekat - Igrac1
    igracTima Igrac1;
    //inicijalizujemo obilježja objekta Igrac1
    strcpy_s(Igrac1.imePrezime,"Refet Gojak");
    Igrac1.visina = 1.98;
    Igrac1.plata = 900;
    Igrac1.brojPogodaka = 50;
    //ispis vrijednosti obilježja objekta Igrac1
    cout<<crt<<Igrac1.imePrezime<<crt;
    cout<<"Visina: "<<Igrac1.visina <<" m";
    cout<<"\nPlata: "<<Igrac1.plata<<" Eura";
    cout<<"\nPogodaka: "<<Igrac1.brojPogodaka<<crt;

    //kreiramo objekta Igrac2 i inicijalizujemo njegova
    //obilježja odmah pri kreiranju objekta
    igracTima Igrac2={"Rusmir Baljic",1.86,1000,70};

    //ispis vrijednosti obilježja objekta Igrac2
    cout<<crt<<Igrac2.imePrezime<<crt;
    cout<<"Visina: "<<Igrac2.visina <<" m";
    cout<<"\nPlata: "<<Igrac2.plata<<" Eura";
    cout<<"\nPogodaka: "<<Igrac2.brojPogodaka<<crt;

    system("pause");
}
```



Najvažnije je primijetiti da su vrijednosti obilježja objekta `Igrac1` i `Igrac2` apsolutno nezavisne. Dakle, **objekti su međusobno u potpunosti odvojeni, a struktura `igracTima` služi samo kao kalup kojim se definiše koja će obilježja posjedovati svaki od njenih objekata.**

Sa objektima, odnosno obilježjima objekata, se mogu vršiti i aritmetičke operacije. Kao ilustraciju pomenutog i u narednom primjeru nastavljamo u stilu sportskih timova. Kreirat ćemo program u kome je deklarirana struktura `igracTima`, a koja je malo izmijenjena u odnosu na verziju iz prethodnog primjera.

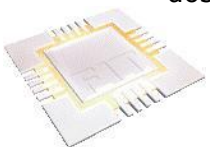
```
#include <iostream>
using namespace std;
char crt[]="\n-----\n";

struct igracTima {
    char imePrezime[40];
    int brojPogodaka;
    double cijena;
};

void main()
{
    //kreiranje tri objekta-instance tipa igracTima
    igracTima Igrac1={"Fuad Dzankovic",30,120000};
    igracTima Igrac2={"Refet Gojak",42,142000};
    igracTima Igrac3={"Rusmir Baljic",25,109500};
    //pravimo kalkulacije nad obilježjima instanci-objekata
    double sumaCijene = Igrac1.cijena + Igrac2.cijena + Igrac3.cijena;
    double prosjekCijene = sumaCijene / 3;
    int sumaPogodaka=Igrac1.brojPogodaka+Igrac2.brojPogodaka+Igrac3.brojPogodaka;
    //ukoliko zelite preciznije informacije promijenite tip podatka u float
    int prosjekPogodaka = sumaPogodaka / 3;
    //ispisujemo stvarne i iskalkulisane vrijednosti
    cout<<"\t\t::TRANSFER IGRACA::"<<crt;
    cout<<"Ime i prezime:\t\t"<<"Golovi:\t\t"<<"Cijena:"<<crt;
    cout<<Igrac1.imePrezime;
    cout<<"\t\t"<<Igrac1.brojPogodaka<<"\t\t"<<Igrac1.cijena<<endl;
    cout<<Igrac2.imePrezime;
    cout<<"\t\t"<<Igrac2.brojPogodaka<<"\t\t"<<Igrac2.cijena<<endl;
    cout<<Igrac3.imePrezime;
    cout<<"\t\t"<<Igrac3.brojPogodaka<<"\t\t"<<Igrac3.cijena<<crt;
    cout<<"SUMA: "<<"\t\t\t"<<sumaPogodaka;
    cout<<"\t\t"<<sumaCijene<<crt;
    cout<<"PROSJEK: "<<"\t\t"<<prosjekPogodaka;
    cout<<"\t\t"<<prosjekCijene<<crt;
    system("pause");
}
```

U prethodnom primjeru je prikazan program koji ispisuje informacije o onim igračima za koje je dogovoren transfer u druge klubove. Na početku programa kreirana je struktura `igracTima` u kojoj su definisana osnovna obilježja igrača neophodna za ovaj program. Nakon toga, u `main` funkciji su kreirana tri objekta tipa `igracTima`, te su im odmah inicijalizovane vrijednosti obilježja. Zatim se u lokalne varijable pohranjuju sumarne i prosječne vrijednosti obilježja `brojPogodaka` i `cijena` za sva tri objekta (igrača), te se na kraju sve to ispisuje u uređenoj tabeli.

Struktura `igracTima` je mogla biti deklarirana unutar `main` funkcije, što apsolutno ne bi utjecalo na izvršenje programa. Međutim, niti jedna druga funkcija osim `main` ne bi mogla koristiti tu strukturu. Znači, kada se struktura deklarira unutar funkcije, ona je dostupna samo toj funkciji, te je niti jedna druga funkcija ili struktura ne može koristiti.



U jeziku C++, korisnički definisanim tipovima se može manipulirati na gotovo identičan način kao i sa ugrađenim tipovima podataka. Zahvaljujući tome, strukture (objekte) možemo kao argumente prosljeđivati funkcijama, a funkcije mogu imati povratnu vrijednost tipa neke strukture. Također, korištenjem operatora dodjele (=) vrijednosti obilježja jednog objekta se mogu dodijeliti drugom objektu. Sve pomenuto je demonstrirano u narednom primjeru:

```
void main()
{
    //prethodni dio koda ostaje isti
    //inicijalizujemo obilježja samo jednog objekta
    igracTima Igrac1={"Fuad Dzankovic",30,120000};

    igracTima Igrac2;//={"Refet Gojak",42,142000};
    igracTima Igrac3;//={"Rusmir Baljic",25,109500};

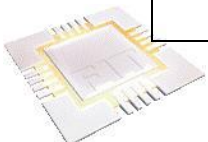
    Igrac2=Igrac1; //<-- dodjela
    Igrac3=Igrac1; //<-- dodjela

    //.... naredni dio koda ostaje isti
    double sumaCijene=Igrac1.cijena+Igrac2.cijena+Igrac3.cijena;
    double prosjekCijene=sumaCijene/3;
    //....
```

Nakon što ste načinili izmjene, ponovo pokrenite program i vidjet ćete da su informacije o sva tri igrača (objekta) potpuno identične.

Kao ilustraciju korištenja objekata u kombinaciji sa funkcijama i dalje se koristi primjer igrača tima. U narednom primjeru su kreirana dva objekta kojima se odmah inicijalizuju vrijednosti obilježja. Nakon toga, objekti se kao argumenti predaju funkciji koja treba da ispiše: informacije o igračima (objektima), te prosječne i sumarne vrijednosti nekih od obilježja.

```
#include <iostream>
using namespace std;
char crt[]="\n-----\n";
struct igracTima {
    char imePrezime[40];
    int brojPogodaka;
    double cijena;
};
//funkcija igracInfo prima dva objekta tipa igracTima
void igracInfo(igracTima Igrac1,igracTima Igrac2){
    double sumaCijene = Igrac1.cijena + Igrac2.cijena;
    double prosjekCijene = sumaCijene / 2;
    int sumaPogodaka=Igrac1.brojPogodaka+Igrac2.brojPogodaka;
    float prosjekPogodaka = (float)sumaPogodaka / 2;
    cout<<"\t\t::TRANSFER IGRACA::"<<crt;
    cout<<"Ime i prezime:\t\t"<<"Golovi:\t\t"<<"Cijena:"<<crt;
    cout<<Igrac1.imePrezime;
    cout<<"\t\t"<<Igrac1.brojPogodaka<<"\t\t"<<Igrac1.cijena<<endl;
    cout<<Igrac2.imePrezime;
    cout<<"\t\t"<<Igrac2.brojPogodaka<<"\t\t"<<Igrac2.cijena<<crt;
    cout<<"SUMA: "<<"\t\t"<<sumaPogodaka<<"\t\t"<<sumaCijene<<crt;
    cout<<"PROSJEK: "<<"\t\t"<<prosjekPogodaka<<"\t\t"<<prosjekCijene<<crt;
}
void main()
{
    //kreiramo dva objekta-instance tipa igracTima
    igracTima Igrac1={"Fuad Dzankovic",30,120000};
    igracTima Igrac2={"Refet Gojak",42,142000};
    //pozivamo funkciju i predajemo kreirane objekte
    igracInfo(Igrac1,Igrac2);
    system("pause");
```



}

U prethodnom programu treba da primijetite način definicije funkcije `igracInfo` iz koga se vidi da funkcija prima dva parametra tipa `igracTima`. Nakon toga, funkcija vrši neophodne kalkulacije i ispis vrijednosti obilježja primljenih objekata. Također, bitno je da uočite način pozivanja funkcije tj. samu predaju argumenata gdje je dovoljno navesti samo ime objekta. Ovaj program radi isto kao i prethodni, samo što se veći dio aktivnosti ne obavlja u `main`, već u funkciji `igracInfo`.

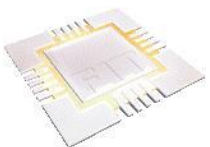
Na početku ovog materijala je pomenuto da obilježja strukture, pored ugrađenih tipova podataka, mogu biti nekog drugog korisnički-definisanog tipa. To znači da u slučajevima kada imamo dvije ili više struktura, obilježja jedne strukture mogu biti tipa neke druge strukture. Kao ilustracija, u nastavku je kreirana još jedna strukturu koja se naziva `Datum` čiji objekat će kao atribut biti dodijeljen strukturi `igracTima`. Pogledajte sljedeći program:

```
#include <iostream>
using namespace std;
char crt[]="\n-----\n";
struct Datum{
    int dan;
    int mjesec;
    int godina;
};
struct igracTima {
    char imePrezime[40];
    int brojPogodaka;
    double cijena;
    //objekat tipa Datum; obiljezja:dan,mjesec,godina
    Datum datumTransfera;
};
void main()
{
    //kreiramo objekat-instancu tipa igracTima
    igracTima Igrac1;
    //inicijalizujemo obiljezja objekta
    strcpy_s(Igrac1.imePrezime,"Refet Gojak");
    Igrac1.brojPogodaka=30;
    Igrac1.cijena=120000;

    //primjetite nacin na koji inicijalizujemo obiljezja
    //objekta datumTransfera koji je tipa Datum
    Igrac1.datumTransfera.dan=12;
    Igrac1.datumTransfera.mjesec=10;
    Igrac1.datumTransfera.godina=2012;

    cout<<"\t\t::TRANSFER IGRACA::"<<crt;
    cout<<"Ime i prezime:\t\tGolovi:\t\tCijena:\t\tTransfer:"<<crt;
    cout<<Igrac1.imePrezime;
    cout<<"\t\t"<<Igrac1.brojPogodaka<<"\t\t"<<Igrac1.cijena<<"\t\t";
    cout<<Igrac1.datumTransfera.dan<<"."<<Igrac1.datumTransfera.mjesec;
    cout<<"."<<Igrac1.datumTransfera.godina<<crt;
    system("pause");
}
```

Obilježje `datumTransfera` pripada strukturi `igracTima`, ali je tipa druge strukture te zbog toga posjeduje sva njihova obilježja (obilježja strukture `Datum`). To se može vidjeti u slučajevima kada se pokuša inicijalizovati jedno od obilježja koje pripada objektu `datumTransfera`, npr:

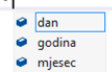



```
//kompajler će prijaviti grešku
Igrac1.datumTransfera = 12;
```

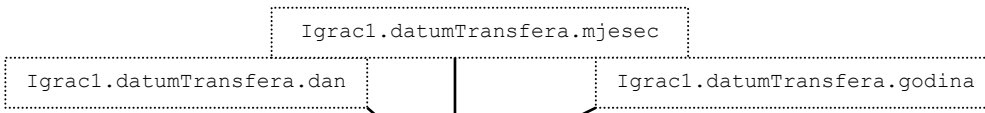
U prethodnom primjeru kompajler će prijaviti grešku iz razloga što nije navedeno obilježje koje se inicijalizuje, pošto znamo da struktura Datum (tj. objekt datumTransfera) ima tri obilježja: dan, mjesec, godina. Upravo iz tog razloga, kod obilježja koja su tipa drugih struktura, odmah nakon znaka tačke, prikazuje se lista obilježja koja su definisana u toj strukturi (što se može vidjeti na slici).

```
Igrac1.brojPogodaka=30;
Igrac1.cijena=120000;

//primjetite nacin na koji ini
//objekta datumTransfera koji
Igrac1.datumTransfera.dan=12;
Igrac1.datumTransfera.
```



U slučajevima kada unutar jednog objekta postoje obilježja tipa druge strukture, inicijalizacija njihovih vrijednosti se može vršiti neposredno pri kreiranju objekta npr:

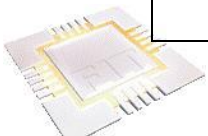


```
igracTima Igrac1={"Refet Gojak", 30, 120000, 20, 10, 2012};
```

Na kraj će biti urađen još jedan primjer u kome korisniku omogućavamo da inicijalizuje vrijednosti obilježja jednog od objekata.

```
#include <iostream>
using namespace std;
char crt[]="\n-----\n";

struct Datum{
    int dan;
    int mjesec;
    int godina;
};
struct igracTima {
    char imePrezime[40];
    int brojPogodaka;
    double cijena;
    Datum datumTransfera;
};
void ispis(igracTima Igrac){
    cout<<Igrac.imePrezime;
    cout<<"\t\t"<<Igrac.brojPogodaka<<"\t\t"<<Igrac.cijena<<"\t\t";
    cout<<Igrac.datumTransfera.dan<<"."<<Igrac.datumTransfera.mjesec;
    cout<<"."<<Igrac.datumTransfera.godina<<crt;
}
void main()
{
    //kreiramo objekat i odmah inicijalizujemo njegova obilježja
    igracTima Igrac1={"Refet Gojak", 30, 120000, 12, 01, 2013};
    //korisniku omogućavamo da inicijalizuje obilježja drugog objekta
    igracTima Igrac2;
    cout<<crt<<"Unesite ime i prezime Igraca 2:";
    //funkciju cin.getline() koristimo u slucajevima kada korisniku želimo
    //omogućiti da inicijalizuje neki niz karaktera. Upravo zbog toga, prvi
    //argument funkcije označava destinaciju na koju se pohranjuje uneseni tekst,
    //nakon čega slijedi maksimalan broj karaktera koji se mogu prihvatiti
    cin.getline(Igrac2.imePrezime, 40);
    cout<<"Unesite broj pogodaka Igraca 2:";
    cin>>Igrac2.brojPogodaka;
    cout<<"Unesite dogovorenu cijenu transfera: ";
    cin>>Igrac2.cijena;
    cout<<"Unesite datum transfera(dd mm gggg): ";
    cin>>Igrac2.datumTransfera.dan;
    cin>>Igrac2.datumTransfera.mjesec;
    cin>>Igrac2.datumTransfera.godina;
    //ispisujemo vrijednosti obilježja za oba objekta
    cout<<crt<<"\t\t\t:TRANSFER IGRACA:"<<crt;
    cout<<"Ime i prezime:\t\tGolovi:\t\tCijena:\t\tTransfer:"<<crt;
    ispis(Igrac1);
```




```
    ispis(Igrac2);  
    system("pause");  
}
```

U cilju provjere znanja o strukturama, prethodnom primjeru dodajte funkciju `unos` koja će omogućiti inicijalizaciju vrijednosti obilježja objekta koji je primljen kao parametar. Dakle, inicijalizacija obilježja objekta se treba vršiti unutar funkcije `unos`, a ne unutar `main` funkcije. Funkciju `unos` kreirajte na način da se unutar `main` funkcije može koristiti na sljedeći način:

```
void main()  
{  
    igracTima Igrac1;  
    igracTima Igrac2;  
    unos(Igrac1);  
    unos(Igrac2);  
    ispis(Igrac1);  
    ispis(Igrac2);  
    //...  
}
```

