Klasa kao realizacija apstraktnog tipa podataka

- Jedna od osnovnih projektnih odluka u dizajniranju jezika C++: korisnički definisane tipove (*user-defined type*) i njihove instance koristiti *što približnije moguće*, ako ne i identično kao i ugrađene tipove (npr. primitivne tipove)
- To znači: *potrebna nam je i semantika kreiranja i kopiranja po vrednosti* i za klase, odnosno njihove objekte
- Tako se notacija i način upotrebe pojednostavljuje:

```
complex c1(-2.5,6.8), c2(246.5,-34.45), c3(0.0,0.0), c4(0.0,0.0);
c3 = complex::add(c1,c2);
c4 = complex::sub(c1,c2);
complex c5 = c3;
c3 = c4;
```

- Ali se semantika izuzetno komplikuje - jedan od najvećih izvora složenosti jezika C++ koju *nemaju* neki drugi, noviji OO jezici, upravo iz ovog razloga
- Potrebno je definisati način inicijalizacije drugim instancama istog tipa, ponovo po vrednosti, kopiranjem:

```
complex c5 = c3; // c3 is not a pointer, but an object
```

- Podrazumevano ponašanje može da bude prosto kopiranje svih podataka članova (što i jeste), ali šta ako je implementacija klase takva da zahteva drugačije, posebno ponašanje u slučaju ovakve inicijalizacije?
- Potreban je *konstruktor kopije* (*copy constructor*) - konstruktor koji se poziva kada se objekat date klase incijalizuje objektom istog tipa; možda ovako:

```
complex::complex(complex other) {
    this->re = other.re;
    this->im = other.im;
}
```

Pristup do člana objekta koji je levi operand operatora ., za razliku od pristupa članu objekta na koga ukazuje pokazivač kao levi operand operatora ->

Klasa kao realizacija apstraktnog tipa podataka

- Prenos objekata kao argumenata poziva funkcija po vrednosti, kopiranjem:

```
complex complex::add (complex ca, complex cb);
```

...`complex::add(c3,c4)`...

- Formalni argument, kao lokalni automatski objekat, inicijalizuje se stvarnim argumentom, na isti način - konstruktorom kopije:

```
complex::complex (complex other);
```

  U trenutku poziva funkcije:

```
complex::add(c3,c4)
```

  na mestu ulaska u funkciju, kreira se lokalni, automatski objekat - formalni argument *ca* (odnosno *cb*) i inicijalizuje stvarnim argumentom *c3* (odnosno *c4*); semantika te inicijalizacije ista je kao i semantika bilo koje druge inicijalizacije, kao da je izvedeno:

```
complex ca = c3;
```

- A kako se vrši ova inicijalizacija? Pozivom konstruktora kopije za objekat *ca* koji se inicijalizuje objektom *c3*:

```
complex::complex(c3)
```

- Međutim, i konstruktor kopije ima formalni argument *other*, koji se opet formira kao automatski objekat i inicijalizuje stvarnim argumentom (*c3*) u trenutku ovog poziva. Kako? Pozivom istog tog konstruktora kopije…
- Problem - beskonačna rekurzija.