Klasa kao realizacija apstraktnog tipa podataka

- Nove operacije koje smo uveli:

```
complex::complex (complex* other) {
  this->copy(other);
}

void complex::copy (complex* other) {
  if (other!=nullptr && other!=this) {
    this->re = other->re;
    this->im = other->im;
  }
}
```

- I onda to koristimo:

```
complex* c1 = new complex(-2.5,6.8);
complex* c2 = new complex(46.5,-34.45);
complex* c3 = complex::add(c1,c2);
complex* c4 = complex::sub(c1,c2);
complex* c5 = new complex(c3);
c3->copy(c4);
…
delete c1; delete c2; delete c3;
delete c4; delete c5;
```

- Problemi:
  - zamorno i nezgrapno (nečitko) manipulisanje objektima i pokazivačima
  - neophodno uništavati objekte (operator *delete*)

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;
}
```