Klasa kao realizacija apstraktnog tipa podataka

- Sledeće pitanje: ako želimo da instance korisničkih tipova koristimo kao i instance ugrađenih tipova, zašto i notacija operacija, recimo za ovakve, matematičke tipove, ne bi bila ista?
- Umesto:

...`complex::sub(complex::add(c3,c4),c5)`...

zašto ne bismo pisali prosto ovako:

...`c3+c4-c5`...

- U skladu sa opredeljenjem na ovakvo korišćenje instanci korisničkih tipova, jezik C++ omogućava i ovo - *preklapanje operatora* (*operator overloading*)
- Umesto "klasičnih" identifikatora, funkcije mogu imati i posebno ime, *operator@*, gde je @ simbol nekog operatora ugrađenog u jezik:

```
class complex {
public:
  complex (double real, double imag);
  complex (complex& other);  // Copy constructor

  friend complex operator+ (complex c1, complex c2);
  friend complex operator- (complex c1, complex c2);

private:
  double re, im;
};

complex::complex (double real, double imag) {…}

complex operator+ (complex c1, complex c2) {
  return complex(c1.re+c2.re,c1.im+c2.im);
}

…
```

Prijateljske finkcije (*friend*) nisu članice date klase, ali imaju pravo pristupa do njenih privatnih

Klasa kao realizacija apstraktnog tipa podataka

- Osim uobičajenog načina poziva preko identifikatora, ovakve *operatorske funkcije* (*operator functions*) mogu se pozivati i implicitno, iz izraza, korišćenjem notacije operatora ugrađenih u jezik:

...`c3+c4-c5`...

- Prevodilac raščlanjuje (parsira, *parse*) izraz na isti način kao što je definisano pravilima jezika i to se ne može promeniti:
  - broj operanada svakog od operatora (*n*-arnost)
  - prioriteti i redosled izračunavanja
  - asocijativnost (način grupisanja - sleva nadesno ili obratno)
- Međutim, ako je neki od operanada instanca korisničkog tipa, poziva se odgovarajuća operatorska funkcija definisana za taj tip operanda, umesto da se generiše uobičajeni mašinski kod za operacije nad ugrađenim tipovima (ponovo polimorfizam!)
- Većina operatora ugrađenih u jezik može se definisati za korisničke tipove (ali ne svi)
- Za mnoge operatore postoji podrazumevano značenje koje se može promeniti (redefinisati)
- Ne mogu se definisati novi operatori (samo redefinisati značenje onih koji su već ugrađeni u jezik)
- Ne može se promenti (redefinisati) značenje za operatore koji rade (samo) nad ugrađenim tipovima (bar jedan operand operatorskih funkcija mora biti korisničkog tipa)
- Mnogo opštih i posebnih pravila za neke posebne operatore - detalji kasnije