Preklopljeni operatori

- Zbog toga je uobičajena praksa da se operatori preklapaju na sledeći način:
  - operatori kod kojih operandi nisu na neki način ravnopravni, simetrični, prvenstveno zbog toga što dati operator ima bočni efekat, odnosno menja svoj operand, definišu se kao nestatičke funkcije članice; na primer, iako operatori složene dodele (+= i ostali) ili operatori ++ i -- mogu, što se tiče jezika, da se definišu i kao funkcije nečlanice, obično se definišu kao nestatičke funkcije članice jer menjaju svoj (levi) operand, pa zato zahtevaju to da taj operand bude baš objekat nad kojim deluju, a ne neka konvertovana vrednost (privremeni objekat):

```
complex complex::operator+= (complex c) {...}

complex c1(3.,4.), c2;
c2 += c1;
```

Tumači se kao *c2.operator+=(c1)*

  - operatori kod kojih su operandi na neki način ravnopravni (simetrični), odnosno koji ne menjaju svoje operande (npr. binarni aritmetički i relacioni operatori, operatori koji rade sa bitima itd.), preklapaju se kao funkcije nečlanice:

```
complex operator+ (complex c1, complex c2) {...}
complex::complex (double d) {...}

complex c1(3.,4.); double d = 5.0; int i = 3;
complex c2 = d+c1, c3 = c2+i, c4 = c2+c3;
```

Preklopljeni operatori

- Operatori ++ i -- imaju prefiksni i postfiksni oblik koji se mogu razlikovati tako što se preklopljeni postfiksni operatori definišu sa parametrom koji prihvata drugi operand tipa *int*; pri pozivu preklopljene operatorske funkcije, ovaj argument ima vrednost 0:

```
complex& complex::operator++ () {...}

complex complex::operator++ (int) {...}

complex c(3.,4.);

c++;

++c;
```

- Preklopljeni operator -> mora biti nestatička funkcija članica koja nema parametre i koja mora da vrati ili običan pokazivač, ili (referencu na) objekat za koji je takođe preklopljen operator ->:

```
template<typename T>
T* smart_ptr<T>::operator-> () {...}

smart_ptr<Clock> p = &clk;

p->tick();
```