Preklopljeni operatori

- Kada traži preklopljeni operator za date tipove operanada, isto kao i za ostale pozive funkcija, prevodilac sprovodi postupak *potrage zavisne od argumenata* (*argument dependent lookup*, *ADL*); pojednostavljeno rečeno, ovaj postupak traži funkciju, uključujući i preklopljen operator, ne samo prema uobičajenim pravilima potrage za nekvalifikovanim imenom u tekućoj oblasti važenja, nego i po klasama kojima pripadaju argumenti (i njihovim osnovnim klasama), ali i po prostorima imena kojima pripadaju te klase
- Ovo omogućava da preklopljeni operatori budu deklarisani u različitim prostorima imena, a da ipak budu odabrani za argumente određenog tipa; na primer:

```
namespace complex {

  class complex {
    …
    friend std::ostream& operator<< (std::ostream& os, const complex& c) {
      return os<<'('<<c.real<<','<<c.imag<<')';
    }

  };

}

int main () {
  complex::complex c1, c2;
  …
  std::cout<<c1<<‘,’<<c2;
}
```

Poziva se preklopljeni operator << za tipove *char* i *double*, definisan u prostoru imena *std*, jer je operand *os* tipa *std::ostream*.
Ovaj izraz vraća referencu na isti objekat klase *ostream* koji je dostavljen kao argument

Poziva se preklopljeni operator << za tip *complex*, definisan u prostoru imena *complex*, jer je operand *c1* tipa *complex::complex*.
Pošto je ovaj operator vratio referencu na objekat tipa *ostream*, može se upotrebiti kao levi operand istog operatora

Preklopljeni operatori

- Jedan aspekt može biti veoma bitan pri odlučivanju o tome da li neki preklopljeni operator treba napraviti kao nestatičku funkciju članicu ili kao funkciju nečlanicu (ako je to moguće):
  - poziv nestatičke funkcije članice za neki izraz ne dozvoljava "promociju" tog izraza, odnosno bilo kakvu implicitnu konverziju levog operanda operatora .: zahteva se da klasa kojoj pripada taj objekat, ili neka njena osnovna klasa, ima navedenu funkciju, inače poziv nije dozvoljen:

```
complex complex::operator+ (complex c) {…}
complex::complex (double d) {…}

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

  - pri pozivu funkcije nečlanice vrše se dozvoljene i definisane implicitne konverzije argumenata; za navedeni primer klase *complex*, ukoliko se operator + definiše kao funkcija nečlanica i definiše navedeni konstruktor konverzije iz tipa *double*, moguće su sve varijante poziva za različite tipove argumenata:

```
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;
```