Kopiranje objekata

- Jedan slučaj u kom se može (pre verzije C++17), odnosno mora (od verzije C++17) vršiti izostavljanje kopiranja jeste inicijalizacija objekta privremenim objektom koji je rezultat izraza, ukjučujući i poziv konstruktora u izrazu: kada se objekat klase *string* inicijalizuje izrazom koji vraća rezultat tipa *string*, za rezultat tog izraza ne mora da se pravi privremeni objekat, već taj rezultat može neposredno da se izgradi u objektu koji se inicijalizuje, tako što se taj objekat inicijalizuje isto kao što bi se inicijalizovao taj privremeni objekat:

```
string sa = s1 + s2;
string sb = string("Hello");
```

- Tada će naredba *return* u pozivu operatorske funkcije *operator+(s1,s2)* konstruisati rezultat u samom memorijskom prostoru objekta *sa* koji se rezultatom ovog poziva operatorske funkcije inicijalizuje, pa konstruktor kopije neće biti pozivan; slično važi i za konstruktor *string("Hello")*
- Ovo važi za sve inicijalizacije objekata, uključujući i parametre funkcija pri pozivu funkcije, kada se oni inicijalizuju stvarnim argumentima koji su rezultati izraza
- Ovo se implementira tako što se funkciji u pozivu zapravo dostavlja, kao jedan od skrivenih parametara, adresa memorijskog prostora za objekat u koji treba da konstruiše i smesti rezultat; ako je to privremeni objekat, na mestu poziva odvaja se taj prostor i pozvanoj funkciji prosleđuje njegova adresa; ako je to objekat koji se inicijalizuje uz izostavljanje kopiranja, onda se dostavlja adresa tog objekta
- Pritom, odgovarajući konstruktor kojim bi se inicijalizovao privremeni objekat, kao i konstruktor kopije moraju biti dostupni kao da se pozivaju (iako se zapravo ne izvršavaju), pa prevodilac sprovodi formalna pravila isto kao da se oni pozivaju, odnosno kao da se ova optimizacija ne vrši (tzv. *as if* pravilo)

Kopiranje objekata

- Kada je privremeni objekat kojim se vrši inicijalizacija operand naredbe *return*, ova ista optimizacija izostavljanja kopiranja naziva se *optimizacija povratne vrednosti* (*return value optimization, RVO*); na primer:

```
string concat (const string& s1, const string& s1) {
  return s1+s2;
}
```

- Moguće su i ovakve višestruke, vezane optimizacije koje izbegavaju višestruka kopiranja
- U opštem slučaju, ukratko, ako je *X* klasni tip, navedene optimizacije izgledaju ovako:

```
X f() {
  return X();
}

f();

X x = X(X(f()));
```

- Do verzije jezika C++17, ovo su bile opcione (neobavezne) optimizacije, iako ih većina prevodilaca sprovodi; od verzije C++17 one su obavezne (uvek se sprovode)
- Programi čija semantika zavisi od toga da li se ove optimizacije sprovode ili ne, odnosno u kojima postoje bočni efekti konstruktora kopije i operatora dodele kopiranjem nisu dobri