Kopiranje objekata

- Prema tome, u ovakvim i sličnim izrazima, vrši se potencijalno mnogo poziva konstruktora kopije ili operatora dodele kopiranjem koji kopiraju sadržaj, tj. alociraju nove pridružene resurse samo da bi kopirali taj isti sadržaj iz privremenih objekata, koji ubrzo potom nestaju
- Kopiranje resursa iz privremenog objekta je prilično beskoristan posao i čist režijski trošak za vreme izvršavanja, pošto privremeni objekat iz kog se pridruženi sadržaj kopira (uz potencijalnu alokaciju novog resursa) ubrzo nestaje, i *sigurno se ne može* koristiti za bilo šta drugo, jer je on nedostupan - on je svakako implicitan rezultat izraza koji se koristi eventualno samo kao operand (argument) naredne operacije i ni za šta više
- Količina kopiranja je utoliko veća ukoliko:
  - prevodilac ne sprovodi optimizaciju izostavljanja kopiranja na svim mestima na kojima je to moguće i dozvoljeno pravilima jezika
  - se parametri u funkcije prenose po vrednosti, a ne po referenci
  - funkcije vraćaju rezultate kopiranjem automatskih objekata
  - se objekti ugrađuju u druge objekte po vrednosti (a ne preko pokazivača), posebno u apstraktne strukture podataka (kolekcije, kontejnere), što je često slučaj, pa se umetanjem u strukturu i vraćanjem vrednosti iz strukture objekti kopiraju (operatorima dodele ili konstruktorima kopije)

Premeštanje resursa

- Opisani problem predstavlja izvornu motivaciju za uvođenje tzv. *semantike premeštanja* (*move semantics*) u jezik C++ (počev od verzije C++11): umesto da se alocirani resursi pridruženi privremenom objektu kopiraju, oni mu se mogu *preoteti*, tj. *premestiti* u objekat koji se tim privremenim objektom inicijalizuje ili kom se taj privremeni objekat dodeljuje, pošto privremenom objektu svakako više nisu potrebni
- Tako je (od verzije C++11) moguće definisati konstruktore i operatore dodele, pa i one koji kao argument mogu da prime objekat istog tipa, koji će se pozivati u slučaju da se kao argument pojavi *dvrednost* (*rvalue*), odnosno rezultat izraza koji nije *lvrednost* (*lvalue*)
- Konstruktor koji prima referencu na dvrednost iste klase naziva se *konstruktor premeštanja* (*move constructor*); nestatička operatorska funkcija članica *operator*= koja prima ovakvu referencu na dvrednost iste klase naziva se *operator dodele premeštanjem* (*move assignment*)
- Parametri koji se vezuju za dvrednosti su tzv. *reference na dvrednosti* (*rvalue reference*) i označavaju se u deklaraciji sa dva znaka *&*:

```
class string {
public:
  …
  string (const string& s);
  string& operator= (const string& s);

  string (string&& s);
  string& operator= (string&& s);
  …
};
```