Kopiranje objekata

- Treba primetiti da ovakva optimizacija ne može da "pređe granice" poziva funkcije (osim eventualno za *inline* funkcije), jer prevodilac u opštem slučaju ne može da zna kako izgleda upotreba parametara u telu funkcije na mestu poziva te funkcije, a parametar mora svakako da se inicijalizuje stvarnim argumentom prilikom poziva. Upravo zato NRVO isključuje parametre, iako su i oni automatski po trajanju skladišta. Na primer:

```
T f (T t) {
  return t;
}

T x;

f(x);
```

Ovde se ne vrši NRVO: konstruktor kopije pozvaće se i za inicijalizaciju parametra stvarnim argumentom (*T t(x)*), a potom i za inicijalizaciju privremenog objekta koji predstavlja rezultat poziva funkcije objektom ja *return (T temp(t))*

- Takođe treba primetiti da se, ukoliko funkcija ima parametre tipa reference, ta referenca samo vezuje za stvarni argument; ukoliko je stvarni argument privremeni objekat, pa zato nije lvrednost, referenca mora biti na konstantu, inače ova inicijalizacija nije dozvoljena. U takvom slučaju, kopiranja svakako nema:

```
T f (const T& t);

f(T());
```

Pošto je parametar funkcije referenca, nema kopiranja stvarnog argumenta

- Ako je parametar klasnog tipa (a ne referenca), kopiranja uvek ima i samo ponekad se može izostaviti:

```
T f (T t);

T x;

f(x);

f(T());
```

Pošto je parametar funkcije klasnog tipa, stvarni argument se kopira u parametar, tj. poziva se konstruktor kopije

Ovde se može (ili mora, za C++17) izostaviti kopiranje

Kopiranje objekata

- Posmatrajmo sada izračunavanje sledećeg izraza, pri čemu su sve imenovane varijable objekti tipa *string*:

```
s = s1 + " " + s2.substr(2,7) + " " + s3.substr(0,2);
```

- Izračunavanje ovog izraza teče ovako (pretpostavka je da je verzija jezika pre C++17):
  - drugi argument poziva *operator+(s1, " ")* je tipa *const char[]*, pa se implicitno konvertuje u *const char** ugrađenom konverzijom ("rastakanje niza u pokazivač"), a potom i korisnički definisanom konverzijom u *string*, pozivom konstruktora konverzije *string(const char*)*; rezultat konverzije je privremeni objekat kojim se inicijalizuje formalni parametar; pošto je to referenca, ona se vezuje za taj privremeni objekat
  - rezultat ovog poziva biće jedan privremeni objekat tipa *string*, označimo ga ovde sa *x*, koji je inicijalizovan povratnim izrazom; taj privremeni objekat ima svoj pridruženi dinamički niz znakova koji sadrži konkatenaciju niza znakova iz *s1* i niza znakova " "; ukoliko prevodilac ne sprovodi *RVO*, ovaj privremeni objekat biće inicijalizovan pozivom konstruktora kopije, ako je operand izvršene naredbe *return* tipa *string*
  - unutar pozvane funkcije *s2.substr(2,7)* pravi se automatski objekat *s* koji alocira prostor za traženi podstring; ovim automatskim objektom se inicijalizuje privremeni objekat koji predstavlja rezultat ovog poziva, označimo ga ovde za *y*; ukoliko prevodilac ne sprovodi *NRVO*, ponovo će se pozivati konstruktor kopije
  - poziva se sada operatorska funkcija za drugi operator + u izrazu; parametri ove funkcije su reference koje se vezuju za navedene privremene objekte, tj. poziva se *operator+(x,y)*

i tako dalje. Na kraju, poziva se operatorska funkcija *operator=* čiji je desni operand privremeni objekat napravljen kao rezultat podizraza sa desne strane znaka =