Kopiranje objekata

- Još jedna varijanta izbegavanja kopiranja jeste i dalje opciona (i u verziji C++17) i naziva se *optimizacija imenovane povratne vrednosti* (*named return value optimization*, *NRVO*): ako je operand naredbe *return* ime (ali ne nepostojanog) automatskog objekta koji nije parametar te funkcije ili parametar *catch* bloka, i koji je istog tipa kao povratni tip funkcije (uz ignorisanje cv-kvalifikacije), zapravo se taj automatski objekat izgrađuje (inicijalizuje) u prostoru objekta koji se vraća iz funkcije; sve operacije nad tim automatskim objektom vrše se u tom memorijskom prostoru. Na primer:

```
string join (const string& s1, const string& s2) {
  string s = s1 + " " + s2;
  return s;
}
```

NRVO: automatski objekat *s* zauzimaće memorijski prostor povratne vrednosti i njegova inicijalizacija i sve druge operacije nad njim vršiće se u tom prostoru

- U opštijem slučaju:

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

NRVO: automatski objekat *t* zauzimaće memorijski prostor povratne vrednosti i njegova inicijalizacija i sve druge operacije nad njim vršiće se u tom prostoru

- Ova optimizacija implementira se na isti opisani način: funkcija prima kao skriveni parametar adresu prostora u koji treba da vrati rezultat; pošto na osnovu koda tela funkcije prevodilac može lako da zaključi da se iza *return* imenuje automatski objekat, sve operacije nad tim objektom (uključujući i poziv konstruktora) može da usmeri na taj memorijski prostor, odnosno tu adresu smatra adresom tog automatskog objekta

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);
```

- 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());
```

- 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());
```