Premeštanje resursa

- Konstruktor premeštanja poziva se kada se objekat inicijalizuje izrazom koji predstavlja dvrednost, a takav je privremeni objekat, pod uslovom da prevodilac ne izostavlja kopiranje:

```
string s = s1 + s2;

void f (string);

f(s1+s2);
```

Ukoliko se ne izostavlja kopiranje, pozvaće se konstruktor premeštanja sa argumentom koji je rezultat izraza *s1+s2*

Ukoliko se ne izostavlja kopiranje, pozvaće se konstruktor premeštanja kojim se inicijalizuje parametar funkcije rezultatom izraza *s1+s2*

- Osim toga, kao specijalan slučaj, konstruktor premeštanja se poziva i kada se povratna vrednost funkcije inicijalizuje imenovanim objektom sa automatskim trajanjem skladišta, ovaj put uključujući i formalni parametar, opet osim ako prevodilac ne vrši NRVO:

```
inline string substr (size_t pos, size_t count) const {
  …
  string s;
  …
  return s;
}
```

Ukoliko se ne vrši NRVO, poziva se konstruktor premeštanja koji inicijalizuje privremeni objekat koji prihvata rezultat poziva funkcije sa argumentom *s*

- Ukoliko klasa nema konstruktor premeštanja, u ovakvim slučajevima pozivaće se konstruktor kopije, pa postojanje konstruktora premeštanja ne treba da menja semantiku programa: konstruktor premeštanja može da popravi performanse i smanji količinu kopiranja resursa, ali pošto njegova aktivacija nije uvek određena (u slučajevima opcionih izbegavanja kopiranja), logika programa ne sme da zavisi od njegovog ponašanja
- Slično važi i za operator dodele premeštanjem: poziva se kada je desni operand dvrednost

Premeštanje resursa

- Pošto većina prevodilaca izostavlja kopiranje čak i kada je to neobavezno, a od verzije C++17 mnoge od navedenih optimizacija postale su obavezne, premeštanje zbog izbegavanja kopiranja resursa iz privremenih objekata gubi na značaju
- Međutim, semantika premeštanja ipak ima svoj značaj (koji postoji oduvek) i koji je možda i važniji od navedenog: premeštanje se može vršiti i na mestima za koje nisu predviđene optimizacije, ukoliko je ono efikasnije i nema potrebe za kopiranjem
- Na primer, bibliotečna šablonska klasa *vector* predstavlja niz promenljivih dimenzija koji se implicitno proširuje po potrebi (operacija *resize*). U tom slučaju za niz elemenata alocira se nov prostor, a elementi vektora se po vrednosti kopiraju na novo alocirano mesto. Ukoliko tip elementa vektora ima semantiku premeštanja, biće upotrebljeno premeštanje umesto kopiranja; ako odgovarajuće operacije premeštanja nisu definisane, vršiće se kopiranje
- Osim toga, neke apstrakcije ne dozvoljavaju kopiranje (jer to nema smisla), ali se njihovi alocirani resursi mogu premeštati; jedan takav primer je apstrakcija ulaznog ili izlaznog znakovnog toka (*istream* i *ostream*)
- U ovakvim situacijama moguće je i eksplicitno zahtevati semantiku premeštanja, iako bi se podrazumevano pozivao konstruktor kopije ili operator dodele kopiranjem: izraz koji je lvrednost se može eksplicitno konvertovati u dvrednost pozivom bibliotečne funkcije *std::move* koja vraća referencu na dvrednost za argument koji može biti i lvrednost; za ovakav rezultat onda se vezuju funkcije čiji parametri primaju reference na dvrednosti, pa i konstruktor premeštanja ili operator dodele premeštanjem:

```
string s1("Hello");
string s2 = std::move(s1);
```