Premeštanje resursa

- Oba problema se mogu prevazići *idiomom kopiranja i zamene* (*copy-and-swap idiom*):
  - ako je desni operand operatora dodele lvrednost, najpre se konstruktorom kopije pravi lokalni automatski objekat - parametar operatorske funkcije, koji kopira resurs argumenta u formalni parametar
  - potom se radi prosta i efikasna razmena (*swap*) resursa između parametra i objekta domaćina (onog na koga ukazuje *this*); ta zamena radi se pomoćnom operacijom *swap* koja ne baca izuzetak (*non-throwing*)
  - kada se izlazi iz ove operatorske funkcije, parametar, kao lokalni automatski objekat biće uništen pozivom destruktora, koji će obrisati prethodni resurs koji je objekat domaćin imao u sebi:

```
inline string& string::operator= (string other) {
  swap(*this,other);
  return *this;
}
```

Funkcija *swap* će samo razmeniti vrednosti pokazivača *this->str* i *other.str*, bez bacanja izuzetka

Objekat *other* se ovde uništava i njegov destruktor briše niz koji je ranije bio pridružen objektu **this*

- Sada će se dešavati sledeće:
  - ako je desni operand operatora dodele dvrednost, npr. privremeni objekat, formalni parametar operatorske funkcije biće inicijalizovan njime i to ili direktnom konstrukcijom, zbog izostavljanja kopiranja (svakako za C++17), ili konstruktorom premeštanja ako ove optimizacije nema:

```
s = string("Hello");
```

Formalni parametar *other* biće inicijalizovan ili pozivom konstruktora konverzije *string(const char*)*, ili pozivom konstruktora premeštanja *string(string&&)* (pre C++17)

  - ako je desni operand operatora dodele lvrednost, formalni parametar operatorske funkcije biće inicijalizovan pozivom konstruktora kopije; ako se pri alokaciji njegovog resursa baci izuzetak, to će biti urađeno pre ulaska u operatorsku funkciju, pa objekat domaćin (levi operand operatora dodele) neće biti promenjen:

```
s1 = s2;
```

Formalni parametar *other* biće inicijalizovan pozivom konstruktora kopije *string(const string&)*

Premeštanje resursa

- Pomoćna operacija *swap* izgleda ovako:

```
friend void swap (string& first, string& second) {
  using std::swap;
  swap(first.str,second.str);
}
```

- Ona je deklarisana kao prijateljska funkcija koja nije članica klase *string*, kako bi bila deklarisana u prostoru imena u kom je i klasa *string*, da bi je postupak potrage po argumentu (*argument dependent lookup, ADL*) pronašao za nekvalifikovanu pretragu u tom opsegu (inače ne bi, da je statička funkcija članica); ovo je samo zato da bi ta funkcija bila dostupna i na drugim mestima za slične potrebe, recimo u apstraktnim strukturama podataka koje treba da rade iste stvari (zamenu vrednosti npr. u svojim *swap* funkcijama koje koriste za premeštanje)
- Treba primetiti da

```
using std::swap;
swap(first.str,second.str);
```

  u opštem slučaju ne znači isto što i:

```
std::swap(first.str,second.str);
```

- Prva varijanta dozvoljava da postupak ADL za argumente poziva funkcije *swap* nađe i funkciju koja bolje odgovara tipovima argumenata a nije u prostoru imena *std*, gde je najpre i traži, ali da ako takvu ne nađe, drugu traži i u prostoru imena *std*. Druga varijanta zahteva da se takva funkcija strogo traži samo u prostoru imena *std*
- Za konkretan slučaj, pošto se radi o prostim pokazivačima tipa *char**, rezultat će biti isti, ali u slučaju drugih, npr. korisničkih tipova, rezultat može biti različit