Objekti sa zauzetim resursima

- Pritom, ove operacije imaju neke zajedničke delove, pa je dobro *refaktorisati* (*refactor*) ovu klasu tako da se ti zajednički delovi izdvoje u pomoćne potprograme i tako eliminišu dupliranja koda:
  - konstruktor kopije radi alokaciju i kopiranje
  - destruktor radi dealokoaciju
  - operator dodele radi najpre dealokaciju (kao destruktor), pa onda alokaciju i kopiranje (kao konstruktor kopije)

```
class string {
public:
  string () : str(nullptr) {}
  string (const char* s) : string() { allocate(s); copy(s); }
  string (const string& s) : string(s.str) {}
  string& operator= (const string& s);

  ~string () { release(); }
  …
protected:
  void allocate (const char* s) { if (s) str = new char[std::strlen(s)+1]; }
  void copy (const char* s) { if (s) std::strcpy(str,s); }
  void release () { delete [] str; str = nullptr; }
  …
};

inline string& string::operator= (const string& s) {
  if (this!=&s) {
    release(); allocate(s.str); copy(s.str);
  }
  return *this;
}
```

Zadatak:
Ova implementacija ima nedostatak zbog toga što operacije *allocate* i *copy* dva puta prolaze kroz ceo izvorni niz znakova (to rade funkcije *strlen* i *strcpy*). Prepraviti je tako da se to radi samo jednom.

Objekti sa zauzetim resursima

- Prema tome, u ovakvim situacijama potreban je posebno definisan konstruktor kopije, operator dodele kopiranjem i destruktor; zato postoji preporuka da se, ako za klasu postoji potreba za jednom od ovih operacija, obrati pažnja i verovatno naprave sve tri, jer je u pitanju možda slučaj objekta sa zauzetim resursima koji se koristi i ugrađuje "po vrednosti"
- Treba primetiti da je sve ovo posledica ključne odluke da se objekti klasa mogu koristiti i ugrađivati "po vrednosti", odnosno da se mogu koristiti na isti način kao i objekti ugrađenih (neklasnih) tipova i biti svih kategorija po životnom veku
- Ukoliko to ne bi bio slučaj, kao što i nije u mnogim drugim novijim objektno orijentisanim jezicima, sve ove komplikacije ne bi bilo; u tim jezicima važi:
  - objekti su samo dinamički i uvek anonimni
  - objektima se pristupa samo preko posrednika (pokazivača, odnosno referenci)
  - postoje samo operacije tih klasa koje se pozivaju eksplicitno; nema operatorskih funkcija
  - nema ugrađenih objekata članova klasa (samo pokazivača/referenci), nema automatskih i statičkih objekata (samo pokazivača/referenci) itd.
  - nema implicitnih kopiranja prilikom inicijalizacije, dodele, prenosa argumenata i povratne vrednosti (kopiraju se i prenose samo pokazivači/reference na objekte)