Objekti sa zauzetim resursima

- Dalje, ako se predviđa korišćenje ovih objekata po vrednosti (svih kategorija životnog veka), korisnici ove klase očekivaće i mogućnost inicijalizacije kopiranjem. Na primer:

```
int main () {
  string s1("Hello"), s2(s1);
  …
}
```

Objekat *s2* inicijalizuje se kopiranjem objekta *s1*

- Prema već navedenim pravilima, ako se u klasi ne navede eksplicitno konstruktor kopije, prevodilac će generisati implicitni konstruktor kopije koji vrši inicijalizaciju kopiranjem objekata članova pozivom njihovih konstrukora kopije; međutim, za članove koji su ugrađenih tipova, vrši se prosto kopiranje vrednosti, što je ovde slučaj, pošto je član *str* pokazivač. Prema tome, u ovom slučaju, objekti klase *string* biće *plitko kopirani* (*shallow copy*), što znači da će se kopirati samo pokazivač, a ne i dinamički niz na koji on ukazuje
- Ovo nije željeno ponašanje, jer će dovesti do toga da kopirani objekti (npr. *s1* i *s2* u primeru gore) dele isti dinamički niz, pa će promene učinjene u jednom biti vidljive i u drugom. Ovde je namera da ti objekti postoje kao nezavisni entiteti i da se njihove izmene rade nezavisno
- Zato nam je ovde neophodan korisnički definisan konstruktor kopije koji vrši *duboko kopiranje* (*deep copy*):

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

 ~string () { delete [] str; str = nullptr; }
  …

};
```

Objekti sa zauzetim resursima

- Potpuno analogno, ako se predviđa korišćenje ovih objekata po vrednosti (svih kategorija životnog veka), korisnici ove klase očekivaće i mogućnost dodele kopiranjem. Na primer:

```
int main () {
  string s1(“Hello”), s2;
  s2 = s1;
  …
}
```

- Prema već navedenim pravilima, ako se u klasi ne navede eksplicitno operator dodele kopiranjem, prevodilac će generisati implicitni operator dodele kopiranjem koji vrši dodelu kopiranjem objekata članova pozivom njihovih operatora dodele kopiranjem; međutim, za članove koji su ugrađenih tipova, vrši se prosto kopiranje vrednosti, što je ovde slučaj, pošto je član *str* pokazivač. Prema tome, i u ovom slučaju objekti klase *string* biće dodelom plitko kopirani

- Zato nam je ovde neophodan i korisnički definisan operator dodele kopiranjem koji vrši duboko kopiranje, ali za razliku od konstruktora kopije, on mora najpre da oslobodi postojeći dinamički niz (ono što radi destruktor), pa onda alocira i kopira novi niz (ono što radi konstruktor kopije):

```
string& string::operator= (const string& s) {
  if (this == &s) return *this;
  delete [] str; str = nullptr;
  if (!s.str) return;
  str = new char[std::strlen(s.str)+1];
  std::strcpy(str,s.str);
  return *this;
}
```