Objekti sa zauzetim resursima

- U mnogim situacijama objekti neke klase imaju zauzete (alocirane), pridružene resurse, kao što su, na primer:
  - dinamički alocirani objekti
  - otvoreni fajlovi
  - drugi resursi alocirani uslugom operativnog sistema ili izvršnog okruženja programa, kao što su npr. memorija, niti, semafori, priključnice (*socket*), neki drugi kanali komunikacije, "ključevi" (*lock*, *mutex*) i sl.
- Ovi resursi po pravilu se alociraju pri inicijalizaciji objekta, što obezbeđuju odgovarajući konstruktori
- Na primer, posmatrajmo klasu *string* koja treba da realizuje apstraktni tip podataka za nizove znakova, za koje želimo da se mogu dimenzionisati dinamički, u trenutku pravljenja, na osnovu potrebe onoga čime se inicijalizuju; osim toga, želimo da implementiramo operacije konkatenacije dva stringa (npr. preklapanjem operatora +) i mnoge druge. Zbog toga nam je potrebno da objekti ove klase imaju pridruženi dinamički niz znakova, jer se samo tako može napraviti niz čija se dimenzija zna tek prilikom pravljenja. Početna skica ove klase:

```
class string {
public:
  string () : str(nullptr) {}
  string (const char*);
  …

private:
  char* str;
};
```

Omogućava podrazumevanu inicijalizaciju:
string s;

Omogućava inicijalizaciju ugrađenim nizovima znakova:
string s("Hello");

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

Objekti sa zauzetim resursima

- Naravno, ove zauzete resurse treba propisno i osloboditi (dealocirati) kada objekat prestaje da živi. Ako se predviđa korišćenje ovih objekata po vrednosti (svih kategorija životnog veka), odnosno ugradnjom (kao podobjekti, npr. objekti članovi), njihovo uništavanje je implicitno. Na primer:

```
int main () {
  string str("Hello");
  …
}
```

- Prema već navedenim pravilima, ako se u klasi ne navede eksplicitno destruktor, prevodilac će generisati implicitni destruktor koji vrši destrukciju objekata članova pozivom njihovih destruktora; međutim, za članove koji su ugrađenih tipova, destrukcija nema nikakvog efekta, što je ovde slučaj, pošto je član *str* pokazivač. Prema tome, u ovom slučaju, dinamički niz znakova neće biti dealociran, pa će postojati problem curenja memorije (*memory leak*)

- Zato nam je ovde neophodan korisnički definisan destruktor koji vrši potrebnu dealokaciju:

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

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

};
```