Konstruktor kopije

- Ako klasa *X* nema nijedan eksplicitno deklarisan konstruktor kopije (tj. korisnički deklarisan konstruktor kopije), prevodilac će implicitno deklarisati konstruktor kopije koji je javan, *inline*, nije *explicit*, i za koji važi:
  - ako svaka direktna i virtuelna osnovna klasa *B* ima konstruktor kopije koji ima parametar tipa *const B&* ili *const volatile B&*, i ako svaka klasa *M* podatka člana ima konstruktor kopije koji prima parametar tipa *const M&* ili *const volatile M&*, onda i ovaj implicitno generisani konstruktor kopije ima parametar tipa *const X&*
  - u suprotnom, ovaj konstruktor kopije ima parametar tipa *X&*
- Klasa može imati i više konstruktora kopije, recimo onaj koji prima *X&* i onaj koji prima *const X&*
- Ako klasa ima neki konstruktor kopije (pa prevodilac ne generiše implicitni konstruktor kopije), programer ipak može forsirati automatsko deklarisanje konstruktora koji bi prevodilac implicitno deklarisao specifikatorom *=default*:

```
struct X {
  X (X&);
  X (const X&) = default;
};
```

- Ako podobjekti osnovnih klasa i objekti članovi ne mogu da se kopiraju, recimo zato što su objekti klasa koje nemaju dostupne konstruktore kopije, ili ako klasa ima korisnički, eksplicitno deklarisan konstruktor premeštanja (*move constructor*) ili operator dodele premeštanjem (*move assignment operator*), onda će ovaj implicitno deklarisani konstruktor kopije biti obrisan (smatraće se da njegov poziv nije dozvoljen, iako je on deklarisan)
- U suprotnom, ako ovaj implicitno deklarisan ili podrazumevani konstruktor kopije nije obrisan, on će biti definisan i vršiće podrazumevano kopiranje podobjekata osnovnih klasa i objekata članova, po istom redosledu kao i u inicijalizaciji; ako su ti podobjekti objekti nekih klasa, pozivaju se njihovi konstruktori kopije, u suprotnom, vrši se prosto kopiranje vrednosti

Copyright 2018-2024 by Dragan Milićev

Destruktor

- Destruktor (*destructor*) je posebna nestatička funkcija članica klase koja se poziva uvek na kraju životnog veka objekta. Svrha destruktora je da oslobodi resurse koje je objekat eventualno zauzimao tokom svog životnog veka
- Destruktor se deklariše kao funkcija članica klase sa imenom te klase i znakom ~ ispred imena klase:

```
struct X {
  ~X ();
  …
};
```

- Destruktor nikada nema parametre, pa klasa ima najviše jedan destruktor (ne može se preklopiti). Destruktor nema povratni tip. Destruktor ima pokazivač *this*, kao i svaka nestatička funkcija članica
- Destruktor se poziva implicitno uvek na kraju životnog veka objekta, bez obzira na taj životni vek:
  - na kraju izvršavanja programa, za statičke objekte
  - po završetku izvršavanja niti, za objekte sa životnim vekom vezanim za nit (*thread local*)
  - po izlasku iz opsega važenja bloka, za automatske objekte napravljene u tom bloku i privremene objekte čiji je životni vek produžen jer je za njih vezana referenca
  - izrazom *delete*, za objekte sa dinamičkim životnim vekom
  - završetkom celog izraza, za privremene objekte napravljene u tom izrazu
  - prilikom razmotavanja steka kod bačenog izuzetka, za sve automatske objekte koji su napravljeni, a čiji se blokovi napuštaju do hvatanja izuetka
- Destruktor se može pozvati i eksplicitno, za objekte koji su napravljeni pomoću operacije *placement new*. Ako se destruktor pozove eksplicitno za objekat čiji se destruktor kasnije poziva i implicitno, ponašanje je nedefinisano

Copyright 2018-2024 by Dragan Milićev