Trajanje skladišta i životni vek

- Svaka *varijabla* (objekat ili referenca) u programu ima svoj *životni vek* (*lifetime*): vreme tokom izvršavanja programa za koje ta varijabla živi i za koje joj se može pristupati:
  - životni vek objekata klasnog tipa i agregata takvih objekata, kao i njihovih podobjekata, počinje nakon što se završi njihova inicijalizacija (poziv konstruktora), osim ako se inicijalizuju tzv. trivijalnim konstruktorom (implicitni konstruktor koji nema baš nikakve efekte za vreme izvršavanja)
  - životni vek objekata klasnog tipa završava se kada započne izvršavanje destruktora (osim ako je destruktor trivijalan - nema baš nikakve efekte)
  - za sve druge objekte (objekte neklasnih tipova, objekte sa trivijalnim konstruktorima i destruktorima, nizovi takvih objekata), životni vek počinje kada se za njih alocira prostor, a završava kada se prostor dealocira
  - životni vek reference počinje kada se završi njena inicijalizacija, a traje kao da je ona skalarni objekat
- Prema tome, iz ovoga sledi:
  - životni vek objekta je isti ili je ugnežđen u vreme trajanja njegovog skladišta (memorijskog prostora)
  - pre početka životnog veka objekta klasnog tipa uvek se (bez ikakvog izuzetka) poziva njegov konstruktor, a nakon završetka životnog veka uvek (bez ikakvog izuzetka) njegov destruktor
  - životni vek referenciranog objekta (na koga ukazuje pokazivač ili upućuje referenca) može biti kraći od životnog veka tog pokazivača ili reference - problem visećih pokazivača/referenci (*dangling reference*)

Trajanje skladišta i životni vek

- Nakon što je alociran prostor za objekat, a pre nego što je započeo njegov životni vek, kao i nakon što je završen životni vek, a pre nego što je dealociran njegov prostor, neke operacije imaju nedefinisane efekte, na primer pristup do nestatičkog podatka člana ili poziv nestatičke funkcije članice tog objekta
- Pojam životnog veka je ortogonalan pojmu opsega važenja:
  - životni vek je koncept vezan za izvršavanje, dok je opseg važenja vezan za prevođenje programa
  - životni vek je vremenski koncept, opseg važenja je prostorni (vezan za deo izvornog koda programa)
- Svaka varijabla ima jednu od sledećih kategorija životnog veka:
  - statički
  - automatski
  - dinamički
  - lokalni za nit (*thread local*)
  - privremeni (*temporary*)
- Jedan od osnovnih principa dizajna jezika C++ je bio taj da objekti svih tipova (i klasnih i neklasnih) mogu biti svih kategorija životnog veka. Ovo je jedan od najvažnijih uzroka koji je doveo do ogromne složenosti ovog jezika; mnoga složena pravila, kao i koncepti, posledica su ove odluke: konstruktori kopije, preklapanje operatora, operator dodele, konstruktor premeštanja, reference, lvrednosti i reference na lvrednosti, dvrednosti i reference na dvrednosti itd.
- Zato je većina drugih, novijih OO jezika krenula drugačijim putem: u njima postoji stroga podela, tako da objekti klasa mogu biti samo dinamički i anonimni, dok instance ugrađenih tipova mogu biti svih drugih kategorija životnog veka i samo oni mogu biti imenovane varijable; i obratno: dinamički mogu biti samo objekti klasa, ostali ne mogu. Ovo je značajno pojednostavilo semantiku tih jezika