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

Trajanje skladišta i životni vek

- Statički životni vek, odnosno statički podaci su najstarija kategorija životnog veka podataka u računarstvu. Oni vode poreklo iz najstarijih programa i programskih jezika, iz vremena kada su programi imali jednostavnu strukturu i zadatak:
  - programi nisu bili interaktivni
  - sve svoje ulazne podatke imali su definisane zajedno sa instrukcijama (naredbama za njihovu obradu) u samom programu
  - program je imao zadatak da te ulazne podatke, definisane statički u okviru programa obradi i ispiše rezultate te obrade (tipično neka matematička, numerička izračunavanja) na izlazni uređaj (tipično linijski štampač)
  - za ovu obradu program je mogao da koristi neke varijable koje su ponovo statički definisane - prostor za njih se unapred alocira, ponovo statički
- Sa pojavom proceduralnog programiranja pojavljue se potreba za lokalnim podacima, čiji je životni vek vezan za aktivaciju potprograma - žive i koriste se samo u tom potprogramu
- Ako ne postoji rekurzija, i ovakvi podaci se mogu alocirati statički (iako su dostupni samo lokalno u potprogramu i iako se reinicijalizuju pri svakoj aktivaciji potprograma), jer u svakom trenutku izvršavanja postoji najviše jedna aktivacija datog potprograma; zato se ti podaci tu mogu adresirati apsolutno (memorijski direktnim adresiranjem)
- Ako postoji mogućnost rekurzije, ovo više nije moguće, jer u datom trenutku može postojati više (i to unapred nepoznat broj) aktivacija istog potprograma, pa se ovakvi podaci moraju alocirati i dealocirati za vreme izvršavanja, i adresirati relativno, tako da se adresiranje u nekoj operaciji unutar potprograma odnosi na trenutno aktuelnu instancu
- Zato se koristi stek: prilikom ulaska u potprogram, na vrhu steka se formira novi blok lokalnih podataka (tzv. aktivacioni blok); operacije u potprogramu adresiraju podatke iz bloka na vrhu steka (relativno adresiranje u odnosu na vrh steka); prilikom povratka iz potprograma, aktivacioni blok se skida sa vrha steka i ponovo postaje aktuelan onaj koji se odnosi na proceduru iz koje je ova pozvana