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

Trajanje skladišta i životni vek

- Razvojem programiranja, a posebno interaktivnih programa i složenih struktura podataka, statički i automatski (lokalni) podaci nisu više dovoljni:
  - nije moguće i nije dovoljno unapred alocirati podatke, statički, jer se ne može znati unapred njihov broj i/ili struktura
  - životni vek vezan za aktivaciju potprograma je previše kratak: potrebno je da podaci nadžive aktivaciju potprograma, ali da ne žive za sve vreme izvršavanja programa
- Zato su potrebni *dinamički objekti*, čije se kreiranje i uništavanje odvija po potrebama programa, odnosno njegove dinamike i logike: takav dinamički objekat može se kreirati u jednom scenariju, u jednom pozivu potprograma, a potom uništiti u nekom sasvim drugom scenartiju, u pozivu nekog drugog potprograma, sve po potrebi logike programa
- Uvođenjem konkurentnog programiranja i pojma niti (*thread*), pojavljuje se i prirodna potreba da se životni vek objekta veže za nit, ali i ne samo to, nego da se jedno isto deklarisano ime vezuje za po jednu instancu u svakoj niti, i da se operacije sa takvim imenom odnose na onu instancu koja je u opsegu te niti (slično kao što se operacije nad lokalnom varijablom vezuju za onu instancu koja se odnosi na tekuću aktivaciju potprograma)