Statički životni vek

- Inicijalizacija statičkih varijabli, odnosno početak njihovog životnog veka razlikuje se za lokalne i ostale statičke varijable
- Statičke varijable koje nisu lokalne (tj. one iz prostora imena ili statički podaci članovi) inicijalizuju se na sledeći način:
  - najpre se vrši *statička inicijalizacija* (*static initialization*), što znači sledeće:
    - *konstantna inicijalizacija* (*constant initialization*) onih varijabli koje su deklarisane kao *constexpr* ili koje su inicijalizovane konstantnim izrazom; po pravilu, ovo prevodilac vrši još za vreme prevođenja (u prevedeni kod upiše vrednosti tih varijabli u statički alociran prostor za njih); čak i ako to ne uradi za vreme prevođenja, mora da obezbedi da se to radi za vreme izvršavanja pre svega ostalog
    - *inicijalizacija nulom* (*zero initialization*): za sve ostale situacije, objekti se inicijalizuju tako da dobiju binarnu vrednost 0, sa značenjem očekivano konvertovane vrednosti (npr. pokazivač će imati *null* vrednost čak i ako se ona ne implementira binarnim nulama)
  - *dinamička inicijalizacija* (*dynamic initialization*) podrazumeva izračunavanje inicijalizatora (kao izraza) za vreme izvršavanja programa, kao i poziv konstruktora objekta klase koji se inicijalizuje, odnosno upis izračunate vrednosti u varijablu koja nije klasnog tipa; ukoliko može, prevodilac dinamičku inicijalizaciju takođe može uraditi za vreme prevođenja

Statički životni vek

- Na primer:

```
int i, *pi;
constexpr double pi = 3.1415926;
double radius = 20.;
inline double area (double r) { return  r*r*pi; }
double a = area(radius);
struct S {
  S () { cout<<"S::S()\n"; }
};
S s;
```

- Dinamička inicijalizacija statičkih varijabli unutar iste jedinice prevođenja obavlja se redosledom njihovih definicija. Redosled inicijalizacije varijabli iz različitih jedinica nije uređen
- Ukoliko se tokom inicijalizacije ovakvih statičkih varijabli dogodi izuzetak, poziva se *std::terminate()*