Statički životni vek

- Sa druge strane, inicijalizacija lokalnih statičkih varijabli je precizno definisana: takva varijabla inicijalizuje se kada kontrola toka prvi put naiđe na njenu definiciju; ako više niti nailazi na tu definiciju, samo prva će izvršiti inicijalizaciju. Svako naredno izvršavanje preskače (ignoriše) tu definiciju
- Ukoliko izvršavanje nikada ne dođe do ovakve definicije, statički objekat neće biti ni inicijalizovan (pa stoga ni uništen pozivom destruktora), na primer ako se funkcija ne pozove ili ne izvrši grana u kojoj je definicija
- S obzirom na svoj statički životni vek, statičke lokalne varijable nadživljavaju izvršavanje bloka (pa i poziv funkcije) u kom su deklarisane, iako imaju lokalni opseg važenja, pa nisu dostupne van te oblasti. Na primer:

```
int a = 1;

void f () {
  int b = 1;
  static int c=1;
  cout<<" a = "<<a++;
  cout<<" b = "<<b++;
  cout<<" c = “<<c++<<'\n';
}

int main () {
  while (a<4) f();
}
```

a je globalni statički objekat: dostupan u celom fajlu, statički inicijalizovan

b je lokalni automatski objekat: inicijalizuje se pri svakom pozivu funkcije

c je lokalni statički objekat: dostupan samo u bloku, inicijalizuje se pri prvom izvršavanju definicije

a i *c* nadživljavaju izvršavanje funkcije, *b* nestaje izlaskom iz funkcije

Ispisaće se:
a = 1 b = 1 c = 1
a = 2 b = 1 c = 2
a = 3 b = 1 c = 3

Statički životni vek

- Zbog svega ovoga, umesto statičkih objekata koji nisu lokalni (npr. globalni ili podaci članovi), bolje je koristiti lokalne statičke objekte, odnosno statičke objekte "umotati" u funkciju (po pravilu nečlanicu ili statičku članicu). Tako se garantuje propisna inicijalizacija, ali i bolja enkapsulacija. Na primer:

```
class Clock {
public:
  Clock (…) { getClockRegister()->add(this); }
  static const ClockRegister* getClocks () { return getClockRegister(); }
private:
  static ClockRegister* getClockRegister ();
};

ClockRegister* Clock::getClockRegister () {
  static ClockRegister clockRegister(…);
  return &clockRegister;
}
```

- Svi statički objekti žive do kraja programa i uništavaju se nakon završetka funkcije *main*. Ako neki statički objekat (npr. lokalni) nije inicijalizovan, neće biti ni uništen (neće biti pozvan njegov destruktor)