Automatski životni vek

- Varijable sa automatskim životnim vekom su lokalne varijable koje nisu označene kao *static*, *extern* ili *thread_local*
- Automatska varijabla inicijalizuje se svaki put kada izvršavanje dođe do mesta njene definicije; svaki nailazak izvršavanja na definiciju pravi novu instancu varijable
- Automatska varijabla se uništava (pozivom destruktora, ako je objekat klase) kada izvršavanje napusti blok u kom je definisana, i to na bilo koji način: prolaskom do kraja bloka, naredbom *return*, ili zbog bačenog izuzetka

```
int f (…) {
  int i = …;
  … f(…) …
  for (i=0; i<10; i++) {
    int j = i+5;
    …
  }
}
```

Automatski objekat: alocira se i inicijalizuje pri svakom pozivu funkcije *f*

Rekurzija: ugnežđeni poziv kreiraće novi skup svojih automatskih varijabli

Automatski objekat: *j* se kreira i inicijalizuje u svakoj iteraciji ove petlje

j prestaje da živi

i prestaje da živi

- Za svako *izvršavanje* definicije ovakvog objekta postoji posebna instanca za vreme izvršavanja; zato prevodilac ne može alocirati prostor za takve objekte u prevedenom fajlu statički, pošto može biti više nezavršenih aktivacija datog bloka (rekurzija ili više niti); zato se ovakvi objekti alociraju na steku

Automatski životni vek

- Parametri funkcije jesu lokalne, automatske varijable koje se inicijalizuju stvarnim argumentima u trenutku poziva te funkcije:

```
void f (X x1) {...}

void g () {
  X x2;
  ...f(x2)...
}
```

- Pored tradicionalnog oblika *for* naredbe kao na jeziku C, jezik C++ ima još jedan oblik ove naredbe, za iteriranje u opsegu (*range*) neke sekvence:

```
int a[] = {0, 1, 2, 3, 4, 5};
std::vector<int> v = {0, 1, 2, 3, 4, 5};

void f () {
  for (int i : a) { cout<<i; }
  for (int& i : a) { i++; }

  for (auto i : v) { cout<<i; }
  for (auto& i : v) { i++; }
}
```

- Katakteristike ovog oblika naredbe *for* su:
  - tip deklarisane varijable treba da bude tip elementa sekvence (uređene kolekcije) kojom se inicijalizuje ili referenca na taj tip; najčešće se koristi *auto* ili *auto&*
  - sekvenca *range* kroz koju se iterira može da bude niz ili objekat klase koja ima članove sa imenom *begin* i *end*
  - petlja iterira kroz dati niz (mora biti poznate dimenzije), ili počev od *range.begin()* do *range.end*()
  - u svakoj iteraciji petlje deklarisana varijabla inicijalizuje se tekućim elementom sekvence