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)...
}
```

Prilikom poziva funkcije *f* parametar *x1* inicijalizuje se kao automatska varijabla: *X x1=x2,* kakav god da je tip *X*

- 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++; }
}
```

i je objekat tipa *int* koji se inicijalizuje (kopijom vrednosti) svakog elementa

i je referenca na objekat tipa *int* koji se inicijalizuje tekućim elementom

- 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

Automatski životni vek

- Kada izvršavanje napušta blok na bilo koji način (prolaskom kroz kraj bloka, naredbom *return* ili zbog bačenog izuzetka), propisno se uništavaju svi automatski objekti tog bloka koji su kreirani, ali i samo oni: ako neki objekat nije kreiran (recimo zato što je pre izvršavanja njegove definicije bačen izuzetak), objekat neće biti ni uništen
- Ovo važi i za napuštanje bloka zbog podignutog izuzetka: svi objekti koji su kreirani, a čiji se blokovi napuštaju do ulaska u odgovarajući *catch* blok, propisno se uništavaju (pozivom destruktora); ovo obuhvata i okružujuće blokove, odnosno blokove funkcija koje su pozvane, a nisu završene
- Analogno važi i za objekte koji su samo delimično kreirani, jer je izuzetak podignut tokom njihove inicijalizacije (poziva konstruktora): svi njihovi podobjekti osnovnih klasa i članovi koji su kreirani biće propisno uništeni, a oni koji nisu, neće
- Ovaj postupak naziva se *razmotavanje steka* (*stack unwinding*):

```
void f () {
  try {
    X x1;
    g();
  }
  catch (…) {}
}

void g () {
  X x2;
  h();
}

void h () {
  X x3;
  thow 0;
}
```

- Automatski objekti se uništavaju po redusledu uvek tačno obrnutom od onog kojim su kreirani