Konstruktor

- Konstruktor se ne nasleđuje, u sledećem smislu: ako osnovna klasa ima konstruktor sa određenim parametrima, ne znači da i izvedena klasa implicitno ima (nasleđuje) takav konstruktor, odnosno da se i objekti te izvedene klase mogu inicijalizovati istim argumentima, već takav konstruktor mora eksplicitno da se definiše u izvedenoj klasi, ako je potreban
- Međutim, konstruktori se mogu i naslediti (*inheriting constructors*), upotrebom direktive *using* u izvedenoj klasi: ako se u definiciji izvedene klase *Derived* navede direktiva *using Base::Base*, gde je *Base* direktna osnovna klasa, onda se u opseg potrage za konstruktorima pri inicijalizaciji objekta izvedene klase uvode svi konstruktori te osnovne klase; ako se pri inicijalizaciji objekta izvedene klase odabere neki od tih konstruktora, on će inicijalizovati podobjekat te osnovne klase, dok će objekti članovi te izvedene klase i ostale njene osnovne klase biti inicijalizovani podrazumevanom inicijalizacijom; pritom, konstruktor izvedene klase sakriva ovakav nasleđeni konstruktor sa istim potpisom. Na primer:

```
struct Base {
  Base (int, int);
  Base (const char*);
};

struct Derived : Base {
  using Base::Base;
  Derived (const char*);
};

int main () {

  Derived d1(1,2);

  Derived d2("Hello");
}
```

U redu: podobjekat *Base* unutar *d1* se inicijalizuje pozivom *Base::Base(int,int)*

Objekat *d2* se inicijalizuje pozivom *Derived::Derived(const char*)*

Konstruktor

- U definiciji konstruktora klase, pre njegovog tela, a iza liste parametara i znaka :, može se navesti *lista inicijalizatora članova* (*member initializer list*), u kojoj se navode inicijalizacije objekata članova i podobjekata osnovnih klasa razdvojene zarezima. Ove inicijalizacije završavaju se pre ulaska u telo konstruktora klase
- Ako neki podobjekat nema podrazumevanu inicijalizaciju, ovde se mora navesti njegova inicijalizacija, u suprotnom, prevodilac će prijaviti grešku. Na primer:

```
struct Base {
  Base (int, int);
};
struct Derived : Base {
  Base b;
  int& r;
  Derived ();
};
Derived::Derived () {}
```

- Ako nestatički podatak član ima podrazumevani inicijalizator u definiciji klase, a naveden je u ovoj listi, biće inicijalizovan kao što piše u toj listi u konstruktoru, a podrazumevana inicijalizacija biće ignorisana:

```
struct X {
  int i = 1;
  X ();
};
X::X () : i(2) {}
```

- Izuzeci tokom ove incijalizacije mogu se hvatati *try-catch* konstruktom na nivou cele funkcije