Konstruktor

- Konstruktor (*constructor*) je specijalna nestatička funkcija članica klase koja se koristi za inicijalizaciju objekata te klase
- Konstruktor može imati proizvoljne parametre, pa klasa može imati više konstruktora. Postupak izbora konstruktora koji će biti pozvan na mestu inicijalizacije određuje se po istim pravilima kao i za ostale preklopljene funkcije, na osnovu tipova argumenata u inicijalizaciji
- Konstruktor nema povratni tip (čak ni *void*)
- Konstruktor ima pokazivač *this*, kao i svaka druga nestatička funkcija članica
- Kao i svaka druga funkcija članica, konstruktor može biti javan, zaštićen ili privatan. Na mestu inicijalizacije, prevodilac proverava dostupnost konstruktora koji je odabran. Prema tome, pošto se pri inicijalizaciji objekta izvedene klase inicijalizuje i podobjekat osnovne klase pozivom konstruktora, ako su svi konstruktori neke klase privatni, iz te klase se ne može izvesti klasa za koju se mogu praviti objekti ili podobjekti
- Konstruktor ne može biti virtuelan, niti može biti cv-kvalifikovan: cv-kvalifikacija nekog objekta stupa na snagu kada je u potpunosti završena njegova inicijalizacija
- Konstruktor zapravo nema ime i ne može se pozvati eksplicitno - on se uvek poziva implicitno za potrebe inicijalizacije kompletnog objekta ili podobjekta
- Ako se tokom izvršavanja tela konstruktora ili u inicijalizaciji podobjekata pozove virtuelna funkcija tog objekta, pozvaće se implementacija te funkcije koja pripada toj klasi čiji se konstruktor izvršava. Ovo je stoga što generisani kod za svaki konstruktor postavlja vrednost VTP tog objekta tako da ukazuje na VT baš te klase, pa konstruktor izvedene klase postavlja na svoju VT prepisujući prethodnu vrednost itd.

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");
}
```