Opseg važenja klase

- Unutar definicije klase, odnosno u oblasti važenja klase mogu biti deklarisani drugi tipovi, uključujući i klase. Svi oni se nazivaju *članovima klase* (*class members*)
- Takve ugnežđene klase, osim što su u oblasti važenja okružujuće klase, nemaju nikakve druge posebne veze; takvo ugnežđivanje ne znači nikako ugrađivanje objekata, već samo logičko "pakovanje" klase
- Na primer, za implementaciju liste potrebna je struktura koja predstavlja jedan element liste i skladišti pokazivač(e) na sledeći (i prethodni) takav element liste. Ovakva struktura, dakle, bitna je samo za klasu liste i ni za koga drugog. Štaviše, bitna je samo za njenu implementaciju, pa je logično da bude definisana unutar klase liste, i to kao privatna, da ne bi bila deo njenog interfejsa:

```
template<typename T>
class List {
public:
  List ();

  void put (T);
  …

private:

  struct ListElement {
    ListElement (const ListElement* next);
    …

  };

};

void List::put (T t) { … }

List::ListElement::ListElement (const List::ListElement* next) { … }
```

Ovoj strukturi *ListElement* ne može da se pristupi van opsega važenja klase *List*

Konstruktor ugnežđene klase *ListElement*

Opseg važenja klase

- Tip definisan unutar oblasti važenja klase se koristi upravo u ovakvim sistuacijama, kada je on bitan samo za interfejs ili implementaciju te klase. Umesto ugnežđivanja takvog tipa, mogao bi se on deklarisati i globalno, ali to je lošije nego ugnežđivanje iz sledećih razloga:
  - ugnežđeni tipovi su “logički upakovani” u okružujuću klasu, što povećava razumljivost programa jer naglašava značaj i upotrebu takvog tipa
  - ugnežđeni tipovi pripadaju oblasti važenja klase, pa ne ”prljaju” globalni prostor imena, odnosno ne uzrokuju sukob imena
  - mogu biti enkapsulirani, jer mogu biti privatni ili zaštićeni, ukoliko su bitni samo za implementaciju, a ne i za interfejs klase