Deklaracije

- Slična potreba međusobnog unakrsnog referenciranja postoji i kod klasa: u definiciji jedne može biti potrebna druga i obratno, recimo, kada su objekti tih klasa obostrano povezani (klase su povezane obostrano navigabilnom asocijacijom); tada se ova cirkularna zavisnost može razrešiti navođenjem *deklaracije - najave klase* (*forward declaration*) koja nije definicija:

```
class Lobby;

class Clock {
public:
  Clock (Lobby* owner);
  …
private:
  …
  Lobby* myLobby;
};

class Lobby {
private:
  …
  Clock* myClock[MaxNumOfClocks];
};
```

Deklaracija - najava klase

Za deklarisanje pokazivača nije potrebna potpuna definicija klase, dovoljna je najava

- Nakon ovakve najave klase unapred, klasa se smatra nekompletnim tipom i ne mogu se definisati objekti te klase, ali se mogu definisati pokazivači i reference na tu klasu; međutim, sa objektima na koje upućuju ti pokazivači i reference se ne može raditi ništa, jer nisu poznati (deklarisani) članovi te klase

Deklaracije

- Osim navedenog, ovakva tehnika koristi se i za smanjenje međusobnih zavisnosti između fajlova-zaglavlja: ako za definiciju jedne klase *A* nije potrebna potpuna definicija druge klase *B*, onda u zaglavlje u kom je data puna definicija klase *A* ne treba uključivati celo zaglavlje u kom je definicija klase *B*, jer će to značajno produžiti prevođenje ako se prvo zaglavlje uključuje dalje (što je često slučaj), nego treba postaviti samo deklaraciju - najavu:

```
class B;

class A {
public:
  A(B*);

  void doSomething();
  …

private:
  B* myB;
  …
};
```

```
#include “A.h”
#include “B.h”

A::A(B* aB) : myB(aB) {…}


void A::doSomething() {

  …myB->aFunction()…

}
```