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;
  …
};
```

Fajl A.h: za ovu definiciju klase *A* nije potrebna puna definicija klase *B*, pa nema potrebe uključiti zaglavlje *B.h*

Fajl A.cpp: ovde su potrebne pune definicije obe klase, pa se uključuju oba zaglavlja

```
#include “A.h”
#include “B.h”

A::A(B* aB) : myB(aB) {…}


void A::doSomething() {

  …myB->aFunction()…

}
```

Deklaracije

- Jezik C++ ima mnogo vrsta deklaracija, od kojih se mnoge mogu navoditi unutar bloka (složene naredbe), odnosno unutar tela funkcija; sve takve deklaracije mogu se navoditi i van tela funkcija
- Unutar bloka, ovakve deklaracije imaju istu sintaksnu kategoriju kao i naredbe, odnosno mogu se naći bilo gde u bloku, ne samo na njegovom početku; tako se imena (npr. varijable) mogu uvoditi na mestu na kom su potrebna, ne obavezno pre toga:

```
int main () {
  unsigned n;
  cin>>n;

  long sum = 0;
  for (unsigned i=0; i<n; i++) {
    int x;
    cin>>x;
    sum+=x;
  }

  cout<<sum;

}
```