Pretprocesiranje

- Direktiva *#include* ima za posledicu to da se sadržaj imenovanog fajla umetne u tekući izvorni kod umesto ove direktive (formalno, odmah nakon linije sa ovom direktivom); takav uključen kod se dalje pretprocerisa, potencijalno rekurzivno (jer uključeni fajl može imati druge *#include* direktive)
- Ova direktiva ima dva oblika:
  - *#include <filename>* pretprocesor traži imenovani fajl počev od predefinsanog mesta; uobičajeno je da se ovo koristi za fajlove koji sadrže deklaracije iz standardnih biblioteka koje dolaze uz prevodilac, pa se oni traže počev od direktorijuma u kome se nalaze ti bibliotečni fajlovi koji dolaze sa instalacijom prevodioca
  - *#include "filename"* pretprocesor traži imenovani fajl počev od nekog drugog predefinisanog mesta, npr. od direktorijuma u kome je i tekući fajl ili koreni direktorijum podešen prevodiocu; koristi se za korisničke fajlove koji čine sam program
- Ovaj princip može se iskoristiti za rešavanje opisanog problema:
  - deklaracije elemenata programskog modula koji je dat u jednom fajlu, a čine *interfejs* tog modula (dakle ne *svih* elemenata koji su u tom moduli definisani, već samo onih koje drugi moduli treba da koriste), izdvajaju se u *fajlove-zaglavlja* (*header file*), sa tipičnom ekstenzijom *.h*
  - u fajlovima u kojima je potrebno koristiti te elemente, ali i u fajl u kome je sama definicija (istim modul), deklaracije se uvoze uključivanjem fajla zaglavlja:

```
// A.h
extern int x;

// A.cpp
#include "A.h"
int x = 1;
```

```
// B.cpp
#include "A.h"
void g () {
  ...x++...
}
```

Ako deklaracija iz *A.h* ne odgovara definiciji u *A.cpp*, prevodilac će prijaviti grešku kada bude prevodio *A.cpp*, jer je *x* dva puta deklarisan različito

Pretprocesiranje

- Rezultat je to da prevodilac "vidi" isti kod kao i kada su deklaracije u svim fajlovima u kojima se koriste bile pisane eksplicitno, ručno, kao i ranije, ali je razlika za programera značajna: deklaracije su lokalizovane na jednom mestu, u jednom modulu (tj. u njegovom *.h* i *.cpp* fajlu, koji moraju biti konzistentni), pa su modifikacije lakše i manje podložne greškama
- Posledično, kada se promeni neki *.h* fajl, svi *.cpp* fajlovi koji uključuju taj *.h* fajl moraju ponovo da se prevedu (ovo uključuje i tranzitivne zavisnosti od drugih uključenih *.h* fajlova), što podrazumeva da će prevodilac prevoditi i sve deklaracije u svim direktno ili indirektno uključenim *.h* fajlovima (barem da bi kerirao svoju tabelu simbola)
- Razvojno okruženje za programiranje (*integrated programming environment*, *IDE*) može da pomogne i smanji količinu fajlova koje treba prevoditi selektivnim prevođenjem samo onih *.cpp* fajlova koji su promenjeni, kao i onih koji zavise od promenjenih *.h* fajlova (odnosno uključuju te fajlove, uzimajući u obzir i tranzitivne zavisnosti)
- Ovakav postupak uslovnog prevođenja svih potrebnih fajlova naziva se *make* postupak (ponegde i *build* postupak), a konfiguracioni fajl koji definiše zavisnosti između fajlova *make* fajl
- Bez ozbira na tu mogućnost, programer treba da se trudi da smanji zavisnosti između fajlova, tako što ne uključuje nepotrebne *.h* fajlove u *.cpp* a posebno u druge *.h* fajlove, ali i ne unosi nepotrebne deklaracije, posebno definicije klasa u *.h* fajlove; ovo je posebno važno u velikim projektima koji mogu da imaju na stotine modula