# 2. előadás: Fordítás első lépése → preprocessing

Gyakorlaton volt már szó a fordítás fázisairól.

## Fordítás lépései

1. Előfordítás (preprocessing)
2. Nyelvi fordítás
3. Linkelés (Összeszerkesztés, linker)

# Előfordítás

Csak szöveg átalakítást végez, nem fordít. Szövegrészek kivágása vagy benntartása, feltételektől függően. (, )

Fájlokat tud bemásolni ()

Olyan, mint a Search & Replace. Nem kell neki C++ kód, akár plain text / szöveges fájlokon is tud dolgozni.

Ez hátrány / veszély is.

## Tipikus preprocesszor direktívák

* – Az "input-output stream" fájl beszúrása. Előre definiált helyről.
* – Az aktuális könyvtárhoz képest relatív módon keresi a fájlt.  
  (Include vajon miért tudja a pontos sorszámot visszaadni?)
* – Minden előfordulást a szövegben lecserél -ra.  
  Például tömb méretének definiálásához:   
  Ciklusban az ciklusfeltétel is lecserélődik -ra.  
  Így csak egy helyen kell átírni, ha változtatni szeretnénk a tömb méretét később. Hibák esélye kisebb.  
   esetén lokális változó nem lehet tömbméret. (Mert fordítás időben nem ismert érték a változó.)  
  C++-ban már van konstans () változó, ami kiváltja ezt.
* – Abszolút érték makró. operátort használ. Ezzel az a gond, hogy nem igazi függvény.  
   ekkor még jó.  
   esetén nem mindig működik:  
  Ha , akkor -öt várnánk, de -et ad vissza:   
  Azért szeretnénk mégis használni, mert a függvényhívás költsége magas. Stack-re be kell írni a paramétereket, bázispointert frissíteni kell, stb.  
  Túlzárójelezéssel javítható: de így sem tökéletes.  
  Ma már nem használunk ilyen makrókat. Az inline függvények jobbak.
* Include guard Ezt ma is használjuk, többek között névütközések elkerülése miatt. De főként, hogy ne legyen kétszer beszúrva ugyan az a header fájl. Hogy ne lehessen kétszer beszúrva.  
  Csak akkor van rá szükség, ha több mint függvénydeklaráció van a header fájlunkban. Például osztálydeklaráció esetén érdemes.  
  (*Ritkán használt magyar neve: "Állomány őrszem".*)  
  Példa:

complex**.**h**:**

#ifndef COMPLEX\_\_H

#define COMPLEX\_\_H

class Complex **{**

double re**,**im**;**

public**:**

//...

**}**

#endif

main**.**cpp**:**

#include "complex.h"

#include "equation\_solver.h"

int main**(){**

Complex a**,** b**,** c**;**

//...

**}**

equation\_solver**.**h**:**

#include "complex.h"

Complex solve**(**Complex a**,** Complex b**,** Complex c**);**

* Definíció / preprocesszor szimbólum megszüntetése.  
  Példa:

#if N < 120

//...

#elif N > 160

//...

#else

#endif

* Milyen fordító fordít. C verziójának lekérdezése.  
  C++ már engedi a függvények túlterhelését (function overloading). A c ezt még nem tudta. Minden függvénynév egyedi kell, legyen.  
  Ez akkor jó, ha szeretnénk, hogy programunk fordítható legyen akár c-vel akár c++-szal. Azonos nevű, különböző paraméterezésű funkciók kezelése.  
  Példa:

#ifdef \_\_cplusplus

extern "C" **{**

#endif

void f**(**int**){...}**

#ifdef \_\_cplusplus

**}**

#endif

* … Jobb kommentkezelés. Csak sor elején működik. Vigyázzunk, hogy a COMMENT változó ne legyen definiálva. Miért van erre szükség: Régi c-ben csak /\* ... \*/-szerű komment van. Ez nem ágyazható egymásba. (Nesting)

## Linking, linker (Összeszerkesztés)

-ből lesz az object fájl. vagy .

-ből hasonlóan.

Az object fájlokból (, ) lesz az összeszerkesztett fájl. Ez futtatható fájl.

Könyvárat is lehet írni, ez linkelés után lesz, ami nem futtatható le.

### Statikus linkelés

Mielőtt eljutna a felhasználóhoz, már összeáll az egész program. Linkeléskor mindent feloldunk, minden szükséges könyvtárat belerakunk a futtatható állományba.

Mindenről egyértelműen kiderül, hogy megvan-e. De nekünk kell újra linkelni, ha új verzió jön ki valamelyik könyvtárunkból.

### Dinamikus linkelés

A ilyen komponens. Dynamic linked library

Futási időre hagyjuk a külső hivatkozások feloldásának nagy részét.

Ettől lesz a sok "missing " hiba. De cserébe kisebb lesz az .

Nem kell újrafordítani, ha új dll verzió jön ki.

Lehet "on demand" lusta, lassú feloldás vagy szigorú, csak akkor indul el a szoftver, ha minden hivatkozás él.