C/C++ előfordítója

Nem hagyományos értelemben a C előfordítóját is nevezhetjük a metaprogramozás egyik eszközének, azzal a különbséggel, hogy közvetlenül a forráskódon végez transzformációkat.

Működésének az alapelve nagyon egyszerűnek tekinthető, hiszen egyszerűen szövegbeszúrásokat és szöveghelyettesítéseket végez a forráskódon. Egyszerűségében rejlik ereje is, ugyanis rendkívüli szabadságod ad a programozó kezébe maga az előfordító, de sajnos ez a gyakorlatban több problémát is eredményezhet.

Magát az előfordítót egy különálló nyelvnek is tekinthetjük, ami a C-től független, mivel még a C nyelv feldolgozása előtt feldolgozásra kerül, minden egyes fordításkor. Több feladat is van az előfordítónak, úgymint a fizikailag több sorban lévő, de logikailag egy sornak számító kódok összefűzése, a preprocesszor dírektíváinak tokenekre bontása, megjegyzések törlése a kódból, és a felhasználó által definiált utasítások végrehajtása (szimbólum behelyettesítés, makrók, esetleg feltételes fordítás).

Az include direktíva az előfordító leggyakrabban használt utasítása. A fordító megkeresi a programozó által megadott fájlt és annak a tartalmát egyszerűen bemásolja a fordítás alatt lévő fájl tartalmába:

#include <iostream> // megkeresi az iostream fájlt és annak a tartalmát bemásolja

int main() {

std::cout << "Hello World!" << std::endl;

return 0;

}

Sajnos egyszerűségében rejlik legnagyobb hátránya is, hiszen a külső függőségek kezelését nem elég ilyen alacsony szinten kezelni. Tipikus, hogy egyes külső fájlokat, több helyen is használni szeretnénk. Ahhoz, hogy ezt megtehessük, minden egyes fájlnál hivatkozni kell rájuk. De az előfordító nem tartja számon azt, hogy mely fájlok kerültek már felhasználásra, ezért könnyen előfordulhat az, hogy egy fájlt kétszer vagy annál többször másolja be. Ez a kódnak a duplikációjához vezet, ami pedig később fordítási hibához. Ezt elkerülendően feltételes fordítással oldják meg az ilyen problémákat.

Az alapötlet az, hogy szimbólumok segítségével tartjuk nyilván, hogy az adott adott header fájlt betöltötte-e már a fordító vagy sem. Ha igen, akkor a feltételes fordítást használva, egy üres fájlt adunk vissza neki, ellenkező esetben az eredeti tartalmat:

#include <iostream>

// megnézi a fordító, hogy definiálták-e már az adott szimbólumat

#ifndef PERSON\_H

// ha nem, akkor megtesszük

#define PERSON\_H

// és visszaadjuk a valódi tartalmat

struct Person {

std::string Name;

int Age;

};

#endif

Ez a sok ellenőrzés, illetve a tartalom beillesztése, több ezer fájl esetében komoly fordítási időt emészthet fel, ami nagy céges projektek esetében jelenleg is komoly probléma. Egyes C++ fordítóknál (ilyen pl. a GCC, vagy a Microsoft Visual C++ fordítója) egy új direktíva bevezetésével a pragma once-al próbálkoztak. Mivel nem szabványos, ezért szemantikailag különbözhetnek és a C nyelvvel se kompatibilis:

#pragma once

struct Person {

std::string Name;

int Age;

};