# A C programozás további eszközei

- Feltételes programfordítás
- Include fájlok
- Moduláris programfejlesztés project-tel

Képforrás: https://www.playkettering.org/event/lets-go-legos/

## Feltételes programfordítás

A # kezdetű, előfordítónak szóló parancsok, ún. direktívák alkalmazásával azt is elérhetjük, hogy programunknak csak meghatározott részei fordítódjanak le és kerüljenek be az .EXE programba. Leggyakrabban egy általunk, vagy a programkörnyezet által rögzített szimbólum definiált, vagy definiálatlan voltától függően ágaztatjuk el a programfordítás folyamatát. A szimbólum definiálásának formája:

**#define <SZIMBÓLUM>**

## Feltételes programfordítás

Ezzel a programszöveg feltételes fordításának megadása a következő lehet:

```
… // ez a programrész feltétel nélkül mindig lefordítódik
#ifdef <SZIMBÓLUM>
… // itt található a szimbólum definiáltsága esetén lefordítódó kódrész
#endif
… // ez a programrész feltétel nélkül mindig lefordítódik
```

A feltételes fordítás eszközével elkerülhető az összetettebb programok esetén az a probléma, hogy az #include utasításokkal többször is beépített állomány (pl. stdio.h) fordítási hibát okozzon.

## Include fájlok

Include fájlok deklarálásával mások, vagy saját magunk által korábban elkészített fájlokat emelhetünk be a programunkba a direktíva megadásának helyén. Szokásos alkalmazása a függvény-prototípusokat, típus-, konstans- és változódeklarációkat tartalmazó **.h headerfájlok** megadása, melynek hatására az előfordító a fájlt beszúrja a programba. Alakja:

```
#include <headerfájl-név>
```

Pl.:
```
#include <stdio.h>
#include <stdlib.h>
```

## Include fájlok ..

Másik, ritkább alkalmazása a saját fájlok beszúrásának végrehajtása. Ilyen fájlokban általában a headerfájlok tartalmához hasonló, több modulban is felhasználásra kerülő infor-mációkat adunk meg. Erre a célra az idézőjelek közötti fájlmegadást szokták alkalmazni:

```
#include "fájlnév"
```

Pl.:
```
#include "glob.dat"
#include "comment.txt"
```

## Mooduláris programfejlesztés project-tel

A **project-fájl** a több modulból álló programok készítését támogató, a modulok összefogására és adminisztrálására szolgáló szövegfájl. A moduláris programkészítés indokai között szerepel a *munkamegosztás*, az *információelrejtés*, a *tárgykódú könyvtár-modulok alkalmazásának igénye*, a fájlkapcsolatokból eredően a *fájlváltozások könnyű átvezetésének* igénye.

A project-fájl a programba beépülő állományok felsorolása, mely függvényszerűen megadhatja az állományok egymástól való függését is a változások átvezetésének megkönnyítésére.

## Mooduláris programfejlesztés project-tel …

Pl.:

forras1.c (header1.h, header2.h)

forras2.c

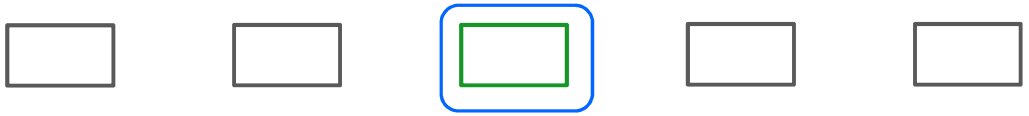
targy3.obj

grafika.lib

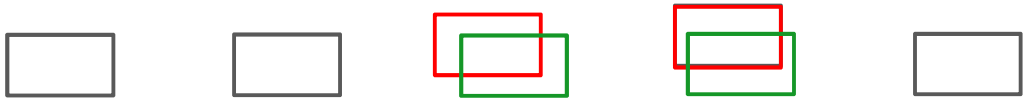
A korszerű C fejlesztőkörnyezetek támogatják a project kezelését: automatikusan létrehozzák, majd módosíthatjuk a .prj kiterjesztésű project-fájlt és a 3 szintű fordítási lehetőséggel kényelmes kezelést tesznek lehetővé.

## Mooduláris programfejlesztés project-tel …

A **Compile** fordítási szint lehetővé teszi az aktív, szerkesztőablakbeli modul tárgykód alakra való fordítását, szintaktikai ellenőrzését.



A **Make** fordítási szint lehetővé teszi a project-leíró fájl információi alapján az összes modul figyelembevételével, lefordításával és összeszerkesztésével az .exe program előállítását. Eközben figyeli a beépülő modulok forrásállo-mányainak módosítási dátumát, és amennyiben azt észleli, hogy valamelyik lefordított állomány forrásállománya időközben módosult, akkor előbb újrafordítja a forrásállományt.



## Mooduláris programfejlesztés project-tel …

A **Build** fordítási szint mellőz minden korábbi fordítási eredményt, tárgykódú állományt és minden forrásállományt újrafordítva hozza létre a projectleíró fájl információit is felhasználva a .exe programot. Erre akkor lehet szükség, ha valamilyen, több modulra kiható változtatást végeztünk a fordítási opciókban, pl. memóriamodellt váltottunk, vagy más optimálási beállításokat akarunk érvényesíteni.