Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

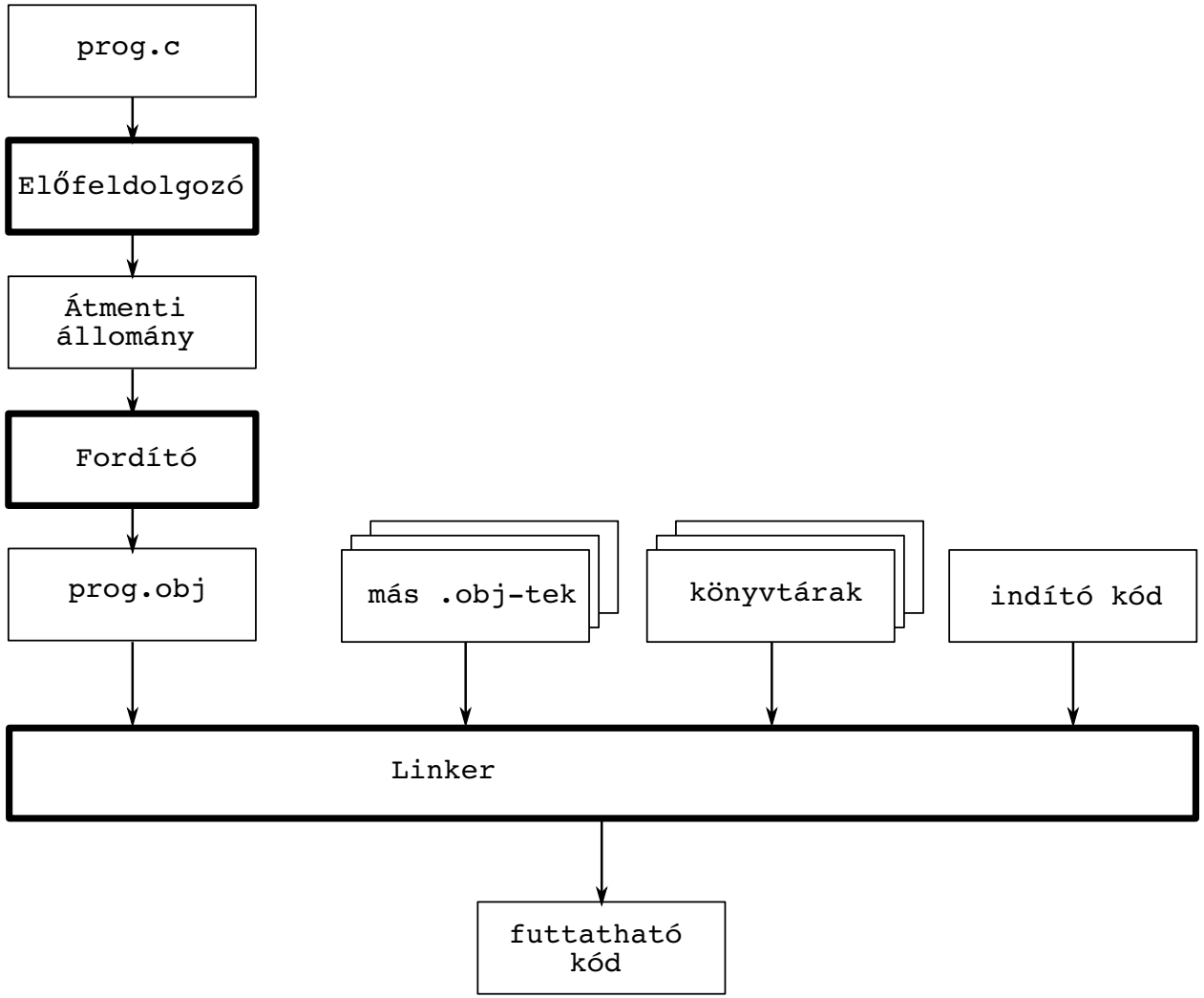
C programozási nyelv
Utasítások III.

Dr. Schuster György

2017. december 28.

A fordítás menete

A fordítás menete



Az fordítás elemei

Az fordítás elemei

előfeldolgozó (precompiler) kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utasításokat.

Az fordítás elemei

előfeldolgozó (precompiler) kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utasításokat.

fordító (compiler) az átmeneti állományból relokálható bináris állományt az un. object kódot állítja elő.

Az fordítás elemei

előfeldolgozó (precompiler) kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utasításokat.

fordító (compiler) az átmeneti állományból relokálható bináris állományt az un. object kódot állítja elő.

linker összeszerkeszti az object állományokat, a használt könyvtárakat és az un. indítókódot. Ezzel futtatható állományt hoz létre.

Az előfeldolgozó utasítások

Az előfeldolgozó utasítások

- minden előfeldolgozó utasítás a # karakterrel kezdődik,

Az előfeldolgozó utasítások

- minden előfeldolgozó utasítás a **#** karakterrel kezdődik,
- az előfeldolgozó utasításokat nem zárja le **;** ,

Az előfeldolgozó utasítások

- minden előfeldolgozó utasítás a **#** karakterrel kezdődik,
- az előfeldolgozó utasításokat nem zárja le **;**,
- az argumentumukat nem tesszük zárójelbe (néhány esetben meg lehet tenni).

`#include` utasítás

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa. Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.
Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.
Az include fájl lehet gyári:

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.
Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.
Az include fájl lehet gyári:

```
#include <stdio.h>
```

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.
Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.
Az include fájl lehet gyári:

```
#include <stdio.h>
```

és lehet általunk előállított:

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.
Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.
Az include fájl lehet gyári:

```
#include <stdio.h>
```

és lehet általunk előállított:

```
#include "enyem.h"
```

`#include` utasítás

Az argumentumában megadott fájlt az adott helyen végigolvassa.
Úgy tekinti az ”include” fájlban lévő sorokat, mintha az a program része lenne.
Az include fájl lehet gyári:

```
#include <stdio.h>
```

és lehet általunk előállított:

```
#include "enyem.h"
```

Van olyan rendszer, amely különbséget tesz a jelölés szerint.

Mi legyen és mi ne legyen az include fájlban

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),
- típus definíciók (typedef)-ek,

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),
- típus definíciók (typedef)-ek,
- struktúra és union definíciók (ez is jön még).

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),
- típus definíciók (typedef)-ek,
- struktúra és union definíciók (ez is jön még).

Ne legyen benne:

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),
- típus definíciók (typedef)-ek,
- struktúra és union definíciók (ez is jön még).

Ne legyen benne:

- változó deklaráció,

Mi legyen és mi ne legyen az include fájlban

Mi legyen benne:

- függvény deklaráció (türelem sorra kerül),
- macro és fordítás idejű konstans (mindjárt jön),
- típus definíciók (typedef)-ek,
- struktúra és union definíciók (ez is jön még).

Ne legyen benne:

- változó deklaráció,
- függvény definíció.

`#define` utasítás

`#define` utasítás

Szerepe kettős:

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni, utána jön a konstans, vagy makró neve,

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,
utána jön a konstans, vagy makró neve,
majd újabb szóköz(ök) és jön, amit helyettesíteni akarunk.

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,
utána jön a konstans, vagy makró neve,
majd újabb szóköz(ök) és jön, amit helyettesíteni akarunk.
Ekkor már lehetnek szóközök is a mezőben.

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,
utána jön a konstans, vagy makró neve,
majd újabb szóköz(ök) és jön, amit helyettesíteni akarunk.
Ekkor már lehetnek szóközök is a mezőben.
A sort a sorvég karakter zárja.

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,
utána jön a konstans, vagy makró neve,
majd újabb szóköz(ök) és jön, amit helyettesíteni akarunk.
Ekkor már lehetnek szóközök is a mezőben.
A sort a sorvég karakter zárja.
Ha nem férünk ki az utolsó mezőnél egy sorban, akkor az utolsó mező \karakterekkel törhető.

`#define` utasítás

Szerepe kettős:

- fordításidejű konstanst definiál,
- makrót definiál.

Szintaktikailag:

```
#define NEV MIT
```

A `#define` után kell legalább egy szóköz jellegű karaternek lenni,
utána jön a konstans, vagy makró neve,
majd újabb szóköz(ök) és jön, amit helyettesíteni akarunk.
Ekkor már lehetnek szóközök is a mezőben.
A sort a sorvég karakter zárja.
Ha nem férünk ki az utolsó mezőnél egy sorban, akkor az utolsó mező \karakterekkel törhető. Pl:

```
#define RNG (a>X && a<X+w && \
             b>Y && b<Y+h)
```

Fordításidejű konstans definálása

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

```
#define NULLA 0
        ⋮
    a=NULLA;
        ⋮
```

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

```
#define NULLA 0
        ⋮
    a=NULLA;
        ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

```
#define NULLA 0
        ⋮
    a=NULLA;
        ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

```
        ⋮
    a=0;
        ⋮
```

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

```
#define NULLA 0
        ⋮
    a=NULLA;
        ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

```
        ⋮
    a=0;
        ⋮
```

Miért jó ez nekünk.

Fordításidejű konstans definálása

Ezt írjuk a forrásba:

```
#define NULLA 0
        ⋮
    a=NULLA;
        ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

```
        ⋮
    a=0;
        ⋮
```

Miért jó ez nekünk.
Mert egyetlen átírással akár több száz helyen is módosíthatunk értéket.

Makró definiálás

Makró definiálás

Ezt írjuk a forrásba:

Makró definiálás

Ezt írjuk a forrásba:

```
#define MAX(a,b)  (a>b)?(a):(b)
          ⋮
     z=MAX(x,y);
          ⋮
```

Makró definiálás

Ezt írjuk a forrásba:

```
#define MAX(a,b)  (a>b)?(a):(b)
          ⋮
     z=MAX(x,y);
          ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

Makró definiálás

Ezt írjuk a forrásba:

```
#define MAX(a,b)  (a>b)?(a):(b)
          ⋮
     z=MAX(x,y);
          ⋮
```

Ez kerül az előfeldolgozás után az átmeneti állományba:

```
          ⋮
     z=x>y?x:y;
          ⋮
```

`#undef` utasítás

`#undef` utasítás

Egy `#define` hatása, vagy az aktuális forrásállomány végéig tart,

`#undef` utasítás

Egy `#define` hatása, vagy az aktuális forrásállomány végéig tart, vagy egy az adott szimbólumra vonatkozó `#undef` utasításig.

`#undef` utasítás

Egy `#define` hatása, vagy az aktuális forrásállomány végéig tart, vagy egy az adott szimbólumra vonatkozó `#undef` utasításig.

Példa:

```
#define NULLA 0
    ⋮          Itt érvényes
#undef NULLA
    ⋮          Itt már nem érvényes
```

`#undef` utasítás

Egy `#define` hatása, vagy az aktuális forrásállomány végéig tart, vagy egy az adott szimbólumra vonatkozó `#undef` utasításig.

Példa:

```
#define NULLA 0
    ⋮          Itt érvényes
#undef NULLA
    ⋮          Itt már nem érvényes
```

Az `#undef` mind a fordítás idejű konstansokra, mind a makrókra alkalmazható

A feltételes fordítás

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy programrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy programrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy programrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy programrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,
- hardver független kódot szeretnénk írni,

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,
- hardver független kódot szeretnénk írni,
- mást szeretnénk a fejlesztési verzióban, mint az éles futó programban,

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,
- hardver független kódot szeretnénk írni,
- mást szeretnénk a fejlesztési verzióban, mint az éles futó programban,
- stb.

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,
- hardver független kódot szeretnénk írni,
- mást szeretnénk a fejlesztési verzióban, mint az éles futó programban,
- stb.

A feltételes fordítás vezérlésére szolgálnak a következő utasítások:
`#if, #else, #elif, #ifdef, #ifndef, #endif`

A feltételes fordítás

A feltételes fordítás fogalma azt jelenti, hogy egy progranrészlet a fordítási folyamat során átkerül-e a forráskódból az átmeneti állományba, vagy nem kerül át.
Normál esetben természetesen - alaposan megdolgozva - átkerül.
Azonban kiválasztott kódrészletek esetén ez vezérelhető.
Miért használunk feltételes fordítást:

- platfrom független kódot szeretnénk írni,
- hardver független kódot szeretnénk írni,
- mást szeretnénk a fejlesztési verzióban, mint az éles futó programban,
- stb.

A feltételes fordítás vezérlésére szolgálnak a következő utasítások:
`#if, #else, #elif, #ifdef, #ifndef, #endif`
Figyelem mindent feltételes fordítási blokkot **`#endif`** utasítás zár le!!

#if, #else, #endif

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk** fordításra kerül.

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk** fordításra kerül.

Ha a **kifejezés** hamis az `#if` és az `#endif` között bármi lehet (még gyerek vers is).

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk** fordításra kerül.
Ha a **kifejezés** hamis az `#if` és az `#endif` között bármi lehet (még gyerek vers is).

```
#if kifejezés
    ⋮
Feltételes blokk1
    ⋮
#else
    ⋮
Feltételes blokk2
    ⋮
#endif
```

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk** fordításra kerül.

Ha a **kifejezés** hamis az `#if` és az `#endif` között bármi lehet (még gyerek vers is).

```
#if kifejezés
    ⋮
Feltételes blokk1
    ⋮
#else
    ⋮
Feltételes blokk2
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk1** fordításra kerül.

#if, #else, #endif

```
#if kifejezés
    ⋮
Feltételes blokk
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk** fordításra kerül.
Ha a **kifejezés** hamis az `#if` és az `#endif` között bármi lehet (még gyerek vers is).

```
#if kifejezés
    ⋮
Feltételes blokk1
    ⋮
#else
    ⋮
Feltételes blokk2
    ⋮
#endif
```

Ha a **kifejezés** igaz a **Feltételes blokk1** fordításra kerül.
Ha a **kifejezés** hamis a **Feltételes blokk2** kerül fordításra.

`#elif` utasítás

`#elif` utasítás

Ez tualjdonképpen egy **`#else #if`** tehát egyedül nem használható. Előnye, hogy egy hosszú - többszörös elágazást lehet csinálni a segítségével egy darab **`#endif`**-el.

`#elif` utasítás

Ez tualjdonképpen egy **`#else #if`** tehát egyedül nem használható. Előnye, hogy egy hosszú - többszörös elágazást lehet csinálni a segítségével egy darab **`#endif`**-el.

```
#if kif1
   ⋮
#elif kif2
   ⋮
#elif kif3
   ⋮
#elif kif4
   ⋮
   ⋮
#elif kifn
   ⋮
#endif
```

`#elif` utasítás

Ez tualjdonképpen egy **`#else #if`** tehát egyedül nem használható. Előnye, hogy egy hosszú - többszörös elágazást lehet csinálni a segítségével egy darab **`#endif`**-el.

```
#if kif1
   ⋮
#elif kif2
   ⋮
#elif kif3
   ⋮
#elif kif4
   ⋮
   ⋮
#elif kifn
   ⋮
#endif
```

`#ifdef` és `#ifndef` utasítások

`#ifdef` és `#ifndef` utasítások

Az **`#ifdef KONSTANS`** utasítás akkor tekinthető igaznak, ha a **`KONSTANS`** definiálásra került. Teljesen mindegy, hogy milyen értékkel.

`#ifdef` és `#ifndef` utasítások

Az **`#ifdef KONSTANS`** utasítás akkor tekinthető igaznak, ha a **`KONSTANS`** definiálásra került. Teljesen mindegy, hogy milyen értékkel.

Az **`#ifndef KONSTANS`** utasítás akkor tekinthető igaznak, ha a **`KONSTANS`** nem került definiálásra. Tipikus felhasználása header állományokban, pl.:

`#ifdef` és `#ifndef` utasítások

Az **`#ifdef KONSTANS`** utasítás akkor tekinthető igaznak, ha a **`KONSTANS`** definiálásra került. Teljesen mindegy, hogy milyen értékkel.

Az **`#ifndef KONSTANS`** utasítás akkor tekinthető igaznak, ha a **`KONSTANS`** nem került definiálásra. Tipikus felhasználása header állományokban, pl.:

```
#ifndef NEVHEADER
        #define NEVHEADER 0
             :
        a header fájl érdemi része
             :
#endif
```

`#pragma`, `#warning` utasítás

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.
A `#warning` arra szolgál, hogy fordítás közben a felhasználónak küldjünk üzenetet.

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.
A `#warning` arra szolgál, hogy fordítás közben a felhasználónak küldjünk üzenetet. Pl.:

```
#warning Hello
```

Ekkor a `Hello` üzenet kiírásra kerül a fordítás során.

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.
A `#warning` arra szolgál, hogy fordítás közben a felhasználónak küldjünk üzenetet. Pl.:

```
#warning Hello
```

Ekkor a `Hello` üzenet kiírásra kerül a fordítás során.
A `##` utasítás (inkább operátor) szövegrészletek összefűzését végzi.

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.
A `#warning` arra szolgál, hogy fordítás közben a felhasználónak küldjünk üzenetet. Pl.:

```
#warning Hello
```

Ekkor a `Hello` üzenet kiírásra kerül a fordítás során.
A `##` utasítás (inkább operátor) szövegrészletek összefűzését végzi. Pl.:
a forrásban

```
value##a=5;
```

`#pragma`, `#warning` utasítás

A `#pragma` előfeldolgozó utasítással a fordító programnak tudunk különböző utasításokat adni. Például adott figyelmeztetéseket ki, illetve be tudunk vele kapcsolni.
A `#warning` arra szolgál, hogy fordítás közben a felhasználónak küldjünk üzenetet. Pl.:

```
#warning Hello
```

Ekkor a `Hello` üzenet kiírásra kerül a fordítás során.
A `##` utasítás (inkább operátor) szövegrészletek összefűzését végzi. Pl.:
a forrásban

```
value##a=5;
```

a befordított kódban

```
valuea=5;
```

Nagyon hasznos programgenerátor írásánál.

Kérdések

Mi a szerepe az előfeldolgozónak?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defenciók, struktúra és union defenciók.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defenciók, struktúra és union defenciók.
Mi ne legyen egy include fájlban?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.
Mi a szerepe a `#define` utasításnak?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.
Mi a szerepe a `#define` utasításnak?
Fordítás idejű konstans és makró létrehozása.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.
Mi a szerepe a `#define` utasításnak?
Fordítás idejű konstans és makró létrehozása.
Meddig tart egy `#define` hatása?

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.
Mi a szerepe a `#define` utasításnak?
Fordítás idejű konstans és makró létrehozása.
Meddig tart egy `#define` hatása?
Az aktuális forrás fájl végéig, vagy adott névvel azonosított `#undef` utasításig.

Kérdések

Mi a szerepe az előfeldolgozónak?
Kiszedi a fordítás számára felesleges elemeneket a forrás állományból és végrehajtja az előfeldolgozó utastásokat.
Mi jellemzi az előfeldolgozó utasításokat?
`#`-al kezdődnek és nem zárja le őket `;`
Mi a szerepe az `#include` utasításnak?
Az argumentumában megadott fájlt az adott helyen végigolvassa és úgy tekinti az include fájlban lévő sorokat, mintha az a program része lenne.
Mi legyen egy include fájlban?
függvény deklaráció, macro és fordtás idejű konstans, tpus defiíciók, struktúra és union defiíciók.
Mi ne legyen egy include fájlban?
Változó deklaráció, függvény defiíció.
Mi a szerepe a `#define` utasításnak?
Fordítás idejű konstans és makró létrehozása.
Meddig tart egy `#define` hatása?
Az aktuális forrás fájl végéig, vagy adott névvel azonosított `#undef` utasításig.

Kérdések

Mi a szerepe az `#if` utasításnak?

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.
Mi zár le egy feltételes fordítási blokkot?

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.
Mi zár le egy feltételes fordítási blokkot?
`#endif` utasítás.

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.
Mi zár le egy feltételes fordítási blokkot?
`#endif` utasítás.
Hogyan működik az `#ifndef` tasítás.

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.
Mi zár le egy feltételes fordítási blokkot?
`#endif` utasítás.
Hogyan működik az `#ifndef` tasítás.
Ha az argumentumában lévő konstans nincs definiálva, akkor lép be a feltételes fordítási blokkba.

Kérdések

Mi a szerepe az `#if` utasításnak?
Feltételes fordítási blokkot kezd meg.
Mi zár le egy feltételes fordítási blokkot?
`#endif` utasítás.
Hogyan működik az `#ifndef` tasítás.
Ha az argumentumában lévő konstans nincs definiálva, akkor lép be a feltételes fordítási blokkba.