Óbudai Egyetem
Kandó Kálmán Villamosmérnöki Kar

C programozási nyelv
Moduláris programozás

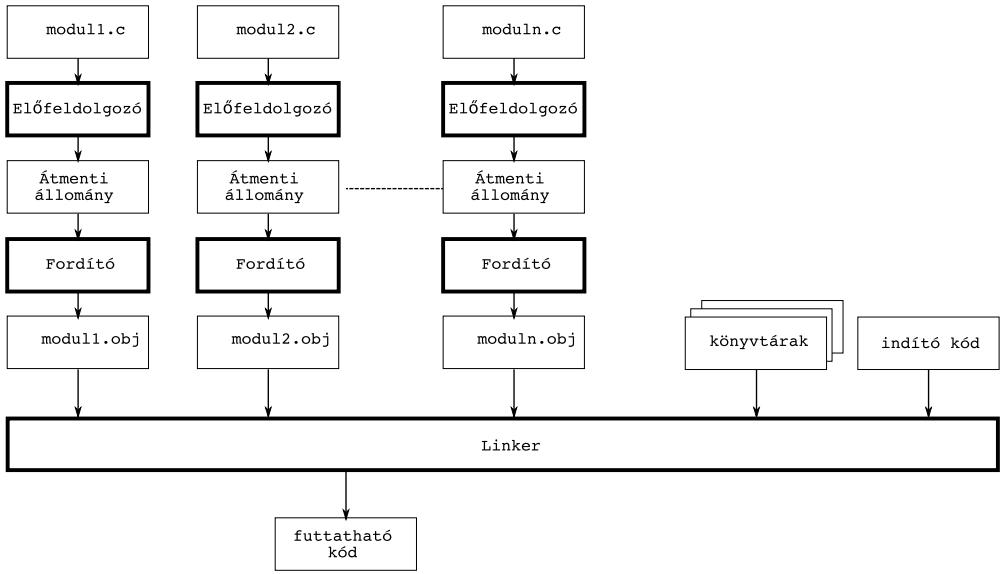
Dr. Schuster György

2017. december 31.

Miért használjuk a moduláris programozást

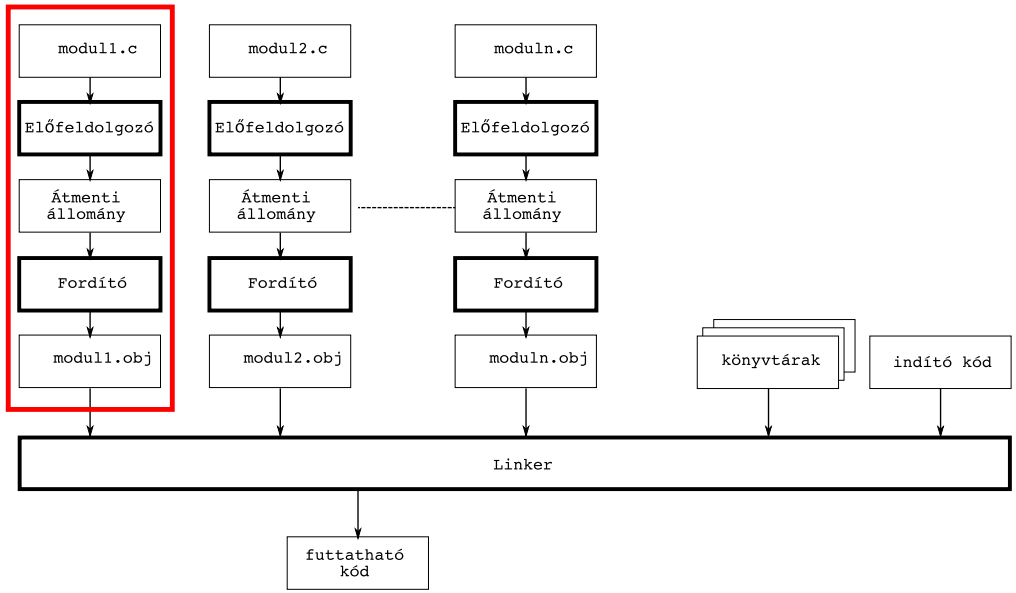
1. a program kódja a mérete miatt kezelhetetlen lenne,
2. több programozó dolgozhat egyidőben az adott feladaton,
3. egy modult könnyebb tesztelni, mint egy teljes programot,
4. egy már megírt és tesztelt modul többször is felhasználható,
5. ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

A fordítás menete (másként)



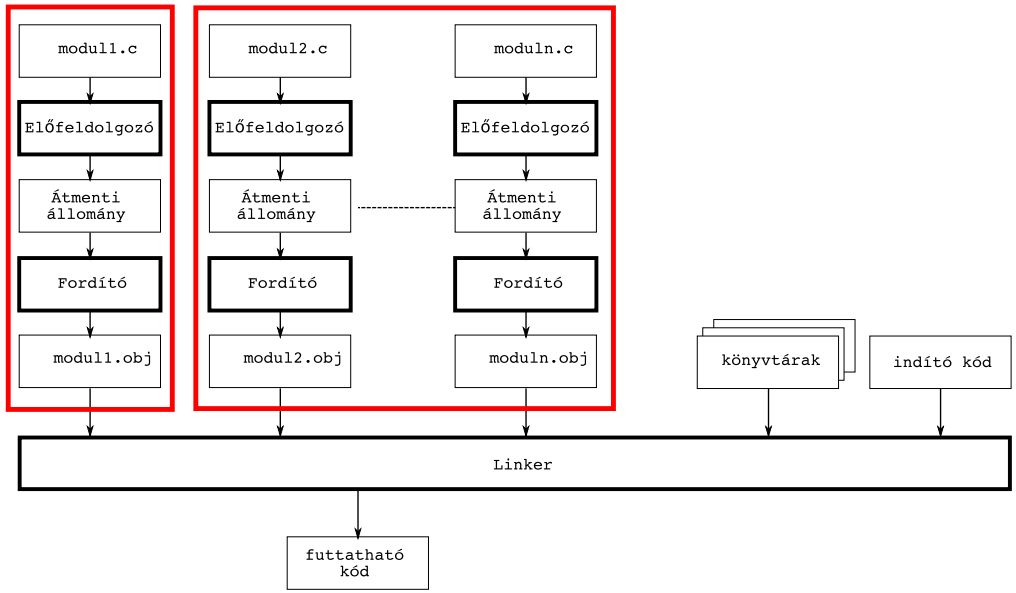
A fordítás menete (másként)

- az aktuális modul lefordul az ismert módon,



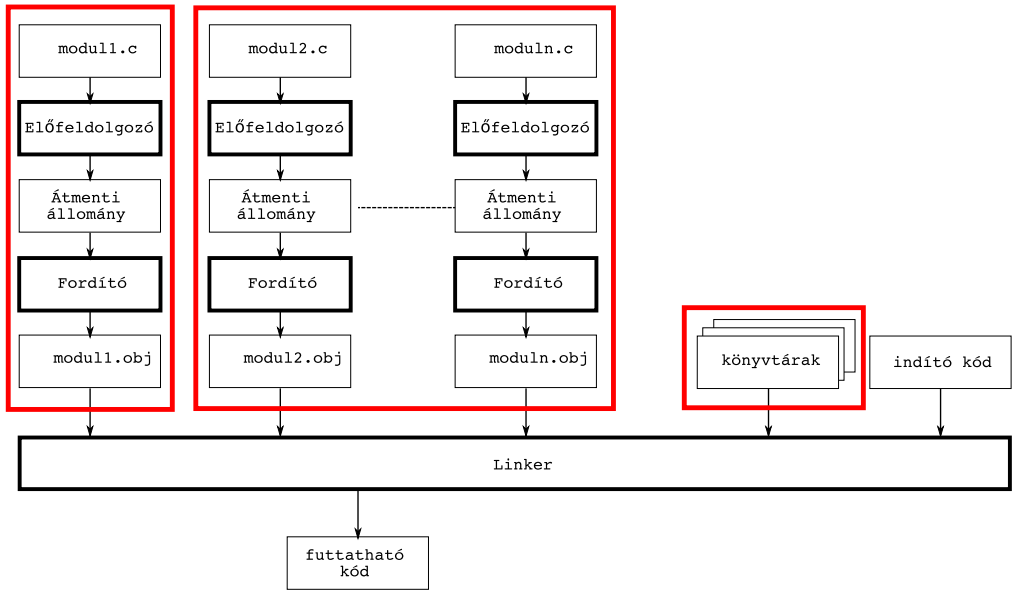
A fordítás menete (másként)

- az aktuális modul lefordul az ismert módon,
- a többi modul is lefordul,



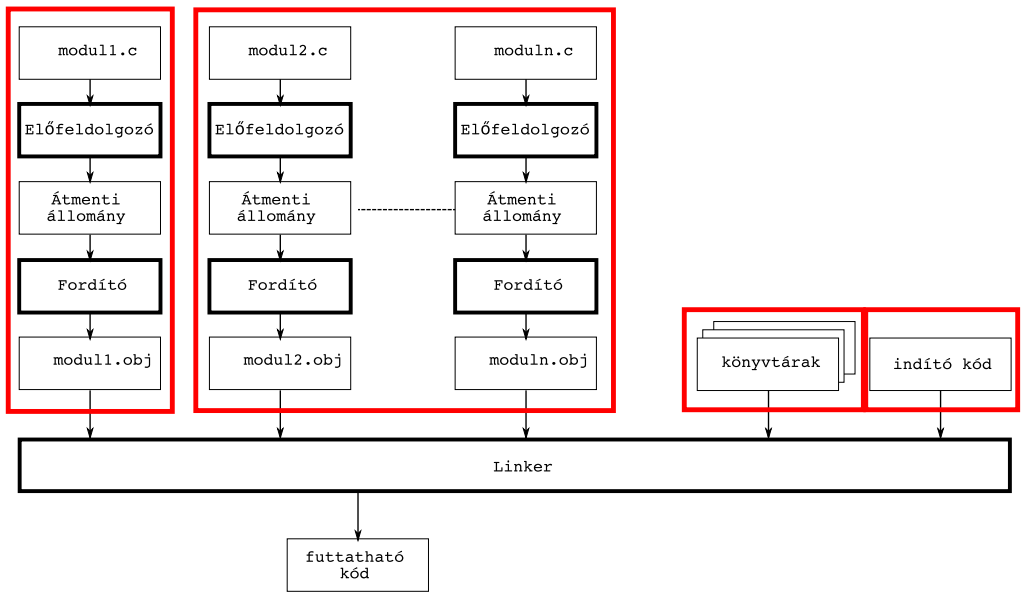
A fordítás menete (másként)

- az aktuális modul lefordul az ismert módon,
- a többi modul is lefordul,
- a könyvtárak is szerepelnek a futtatható kódban,



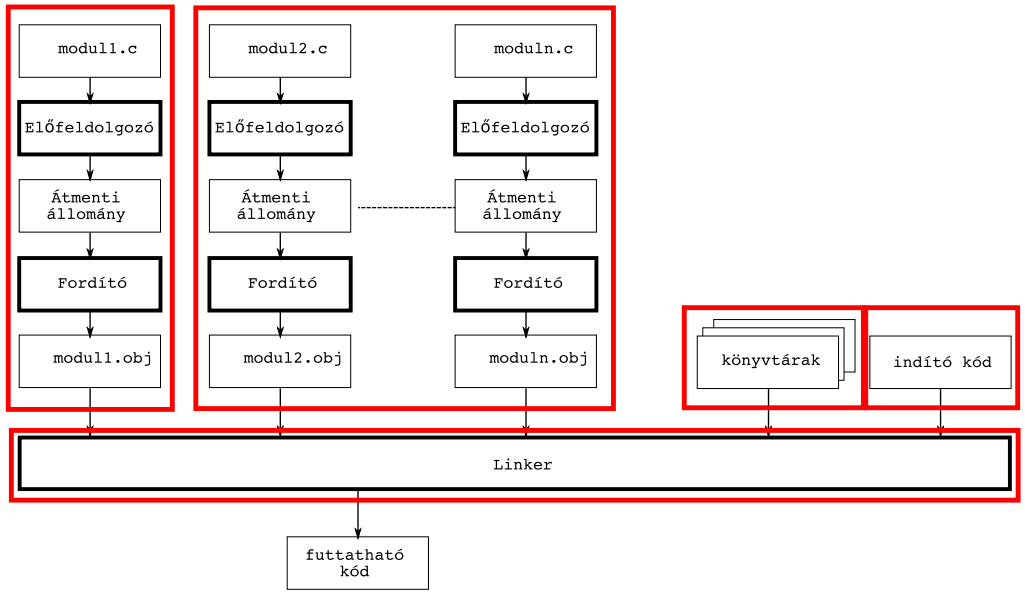
A fordítás menete (másként)

- az aktuális modul lefordul az ismert módon,
- a többi modul is lefordul,
- a könyvtárak is szerepelnek a futtatható kódban,
- az indító kód is szerepel,



A fordítás menete (másként)

- az aktuális modul lefordul az ismert módon,
- a többi modul is lefordul,
- a könyvtárak is szerepelnek a futtatható kódban,
- az indító kód is szerepel,
- a teljes szerkesztést a linker végzi.



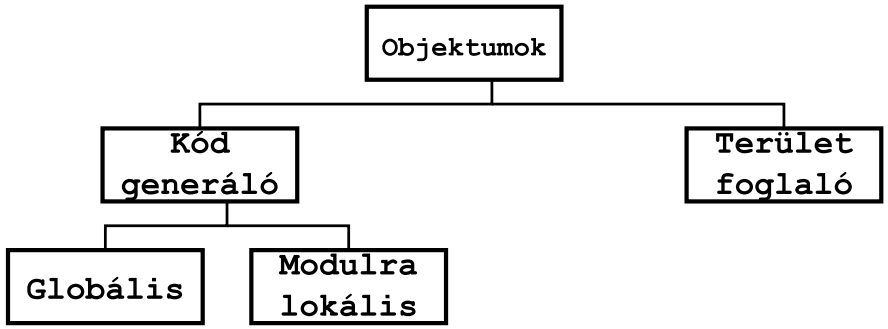
A C objektumai

A C objektumai

- a kódgeneráló objektumok a függvények, a terület foglalók a változók,

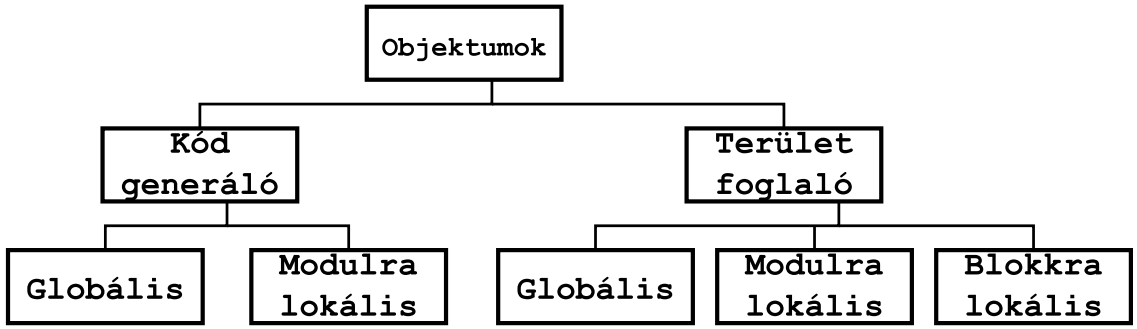
A C objektumai

- a kódgeneráló objektumok a függvények, a terület foglalók a változók,
- egy függvény lehet globális és modulra lokális,



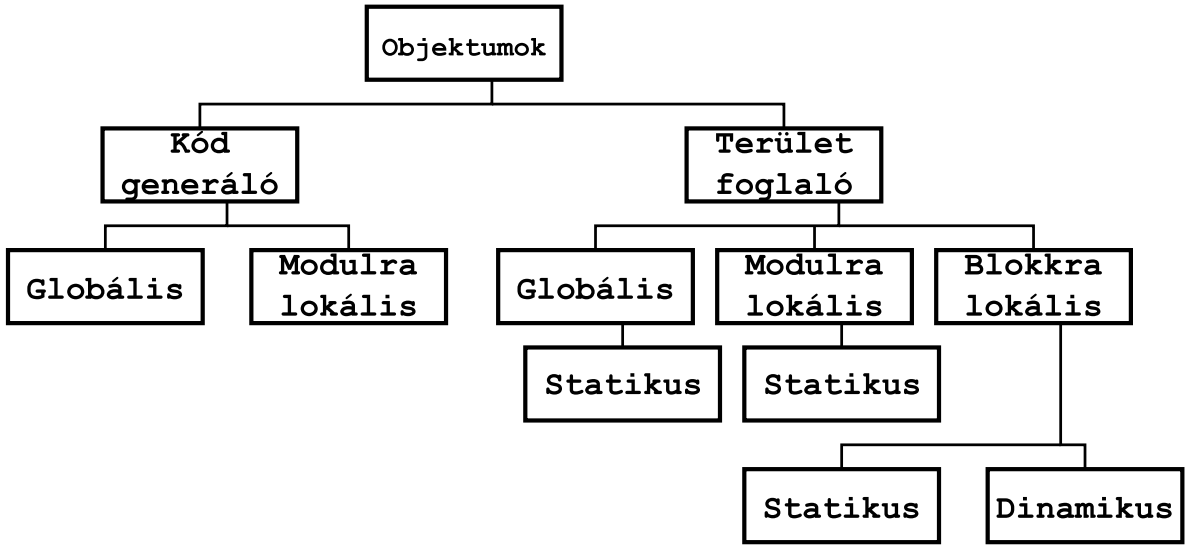
A C objektumai

- a kódgeneráló objektumok a függvények, a terület foglalók a változók,
- egy függvény lehet globális és modulra lokális,
- egy változó lehet globális, modulra lokális és blokkra lokális,



A C objektumai

- a kódgeneráló objektumok a függvények, a terület foglalók a változók,
- egy függvény lehet globális és modulra lokális,
- egy változó lehet globális, modulra lokális és blokkra lokális,
- a változó lehet statikus és dinamikus.



Definíciók

Modul a fordító által lefordítható forrásállomány.

Definíciók

Modul a fordító által lefordítható forrásállomány.

A modulok nem tévesztendők össze az include fájlokkal.

Definíciók

Modul a fordító által lefordítható forrásállomány.

A modulok nem tévesztendők össze az include fájlokkal.

Globális az a függvény, amely minden modulból láthatóvá tehető.

Definíciók

Modul a fordító által lefordítható forrásállomány.

A modulok nem tévesztendők össze az include fájlokkal.

Globális az a függvény, amely minden modulból láthatóvá tehető.

Modulra lokális az a függvény, amely csak az adott modul függvényei számára látható.

Definíciók

Blokkra lokális az a változó, amely csak abban a függvényben látható, ahol deklarálták.

Definíciók

Blokkra lokális az a változó, amely csak abban a függvényben látható, ahol deklarálták.

Modulra lokális az a változó, amely csak az adott modul függvényeiből látható, a kérdéses függvényben azonos nevű változó nincs deklarálva.

Definíciók

Blokkra lokális az a változó, amely csak abban a függvényben látható, ahol deklarálták.

Modulra lokális az a változó, amely csak az adott modul függvényeiből látható, a kérdéses függvényben azonos nevű változó nincs deklarálva.

Globális az a változó, amely minden modulból láthatóvá tehető, ha nincs azonos nevű blokkra lokális és modulra lokális változó.

Definíciók

Blokkra lokális az a változó, amely csak abban a függvényben látható, ahol deklarálták.

Modulra lokális az a változó, amely csak az adott modul függvényeiből látható, a kérdéses függvényben azonos nevű változó nincs deklarálva.

Globális az a változó, amely minden modulból láthatóvá tehető, ha nincs azonos nevű blokkra lokális és modulra lokális változó.

Egy változó statikus, ha a számára lefoglalt hely fordítás időben jön létre.

Definíciók

Blokkra lokális az a változó, amely csak abban a függvényben látható, ahol deklarálták.

Modulra lokális az a változó, amely csak az adott modul függvényeiből látható, a kérdéses függvényben azonos nevű változó nincs deklarálva.

Globális az a változó, amely minden modulból láthatóvá tehető, ha nincs azonos nevű blokkra lokális és modulra lokális változó.

Egy változó statikus, ha a számára lefoglalt hely fordítás időben jön létre.

Egy változó dinamikus, ha a számára lefoglalt hely futás időben jön létre.

Következmények

- Egy blokkra lokális változó kitakar egy azonos nevű modulra lokális és globális változót.
- Egy modulra lokális változó kitakar egy azonos nevű globális változót.
- A statikus változók deklarációnál kezdeti inicializálás nélkül (általában) 0 kezdőértéket vesznek fel.
- A dinamikus változók deklarációnál kezdeti inicializálás nélkül meghatározatlan kezdőértékkel rendelkeznek.

Példa:

Készítsünk programot, amely három modulból áll, ezek: az **1.c**, a **2.c** és a **3.c**.

```
1 #include <stdio.h>
2 void fgv1(void);
3 void fgv2(void);
4 char txt[]="G text\n";
5 int main(void)
6   {
7   fgv1();
8   fgv2();
9   return 0;
10  }
```

1.c

```
1 #include <stdio.h>
2 void fgv1(void);
3 static void fgv2(void);
4 extern char txt[];
5 void fgv1(void)
6   {
7   printf("%s",txt);
8   fgv2();
9   }
10 void fgv2(void)
11   {
12   printf("ML fgv2\n");
13   }
```

2.c

```
1 #include <stdio.h>
2 void fgv2(void);
3 void fgv3(void);
4 static char txt[]="M text\n";
5 void fgv2(void)
6   {
7   printf("G fgv2\n");
8   printf("%s",txt);
9   fgv3();
10  }
11 void fgv3(void)
12   {
13   char txt[]="B text\n";
14   printf("%s",txt);
15   }
```

3.c

Példa:

Láthatjuk, hogy a **`main`** függvény az **`1.c`**-ben található. Egy darab globális **`main`** függvénynek szerepelni kell, de csakis egynek. Az hogy ez melyik modulban van az lényegtelen a progam futása szempontjából.

Példa:

Láthatjuk, hogy a **`main`** függvény az **`1.c`**-ben található. Egy darab globális **`main`** függvénynek szerepelni kell, de csakis egynek. Az hogy ez melyik modulban van az lényegtelen a progam futása szempontjából.

Az **`1.c`** modulban a **`char txt[]="G text\n;"`** változó minden függvényen kívül került deklarálásra (4. sor). Ezért ez a változó egy (inicializált) globális változó.

Példa:

Láthatjuk, hogy a **`main`** függvény az **`1.c`**-ben található. Egy darab globális **`main`** függvénynek szerepelni kell, de csakis egynek. Az hogy ez melyik modulban van az lényegtelen a progam futása szempontjából.

Az **`1.c`** modulban a **`char txt[]="G text\n;"`** változó minden függvényen kívül került deklarálásra (4. sor). Ezért ez a változó egy (inicializált) globális változó.

A **`main`** függvény meghív két függvényt az **`fgv1`**-et (7. sor) és az **`fgv2`**-őt (8. sor). Ezek a függvények az **`1.c`**-ben csak deklarálásra kerültek (2. és 3. sor), de nincsennek definiálva. Ezek a függvények globális függvények.

Példa:

```
1  #include <stdio.h>
2  void fgv1(void);
3  void fgv2(void);
4  char txt[]="G text\n";
5  int main(void)
6    {
7    fgv1();
8    fgv2();
9    return 0;
10   }
```

1.c

```
1  #include <stdio.h>
2  void fgv1(void);
3  static void fgv2(void);
4  extern char txt[];
5  void fgv1(void)
6    {
7    printf("%s",txt);
8    fgv2();
9    }
10 void fgv2(void)
11   {
12   printf("ML fgv2\n");
13   }
```

2.c

```
1  #include <stdio.h>
2  void fgv2(void);
3  void fgv3(void);
4  static char txt[]="M text\n";
5  void fgv2(void)
6    {
7    printf("G fgv2\n");
8    printf("%s",txt);
9    fgv3();
10   }
11 void fgv3(void)
12   {
13   char txt[]="B text\n";
14   printf("%s",txt);
15   }
```

3.c

Példa:

A `2.c` modulban az `fgv1` deklarálásra került (2. sor) és definiálva is van (5-9. sor), előtte semmijen jelző nincs. Tehát ez a függvény egy globális függvény.

Példa:

A **`2.c`** modulban az **`fgv1`** deklarálásra került (2. sor) és definiálva is van (5-9. sor), előtte semmijen jelző nincs. Tehát ez a függvény egy globális függvény.

A harmadik sorban az **`fgv2`** (3. sor) függvény szintén deklarálva van, de a deklaráció egy **`static`** bejegyzéssel kezdődik. A függvény a (10-13. sorban) van definiálva. Ez a függvény egy modulra lokális függvény, tehát csak a **`2.c`** modulban szereplő függvényekből hívható, amint ezt az **`fgv1`** meg is teszi a 8. sorban.

Példa:

A **2.c** modulban az **fgv1** deklarálásra került (2. sor) és definiálva is van (5-9. sor), előtte semmijen jelző nincs. Tehát ez a függvény egy globális függvény.

A harmadik sorban az **fgv2** (3. sor) függvény szintén deklarálva van, de a deklaráció egy **static** bejegyzéssel kezdődik. A függvény a (10-13. sorban) van definiálva. Ez a függvény egy modulra lokális függvény, tehát csak a **2.c** modulban szereplő függvényekből hívható, amint ezt az **fgv1** meg is teszi a 8. sorban.

A **char txt[]="G text\n;"** változó deklarációja kibővült egy **extern** kulcsszóval. Ez azt jelenti, hogy a **char txt[]="G text\n;"** változó egy globális változó, amely számára nem ebben a modulban van a tárolási hely foglalva.

Példa:

```
1  #include <stdio.h>
2  void fgv1(void);
3  void fgv2(void);
4  char txt[]="G text\n";
5  int main(void)
6    {
7    fgv1();
8    fgv2();
9    return 0;
10   }
```

1.c

```
1  #include <stdio.h>
2  void fgv1(void);
3  static void fgv2(void);
4  extern char txt[];
5  void fgv1(void)
6    {
7    printf("%s",txt);
8    fgv2();
9    }
10 void fgv2(void)
11   {
12   printf("ML fgv2\n");
13   }
```

2.c

```
1  #include <stdio.h>
2  void fgv2(void);
3  void fgv3(void);
4  static char txt[]="M text\n";
5  void fgv2(void)
6    {
7    printf("G fgv2\n");
8    printf("%s",txt);
9    fgv3();
10   }
11 void fgv3(void)
12   {
13   char txt[]="B text\n";
14   printf("%s",txt);
15   }
```

3.c

Példa:

A `3.c` modulban az **`fgv2`** minden jelző nélkül deklarálásra (2. sor) és definálásra (5-10. sor) is került. Ezért ez a függvény a globális **`fgv2`**.

Példa:

A `3.c` modulban az **`fgv2`** minden jelző nélkül deklarálásra (2. sor) és definálásra (5-10. sor) is került. Ezért ez a függvény a globális **`fgv2`**.

*A `2.c` modulban is van **`fgv2`** az takarja ezt az **`fgv2`**-t.*

Példa:

A `3.c` modulban az **`fgv2`** minden jelző nélkül deklarálásra (2. sor) és definálásra (5-10. sor) is került. Ezért ez a függvény a globális **`fgv2`**.

*A `2.c` modulban is van **`fgv2`** az takarja ezt az **`fgv2`**-t.*

A 4. sorban a **`char txt[]="M text\n;"`** változó deklarációja előtt ott a már látott **`static`** kulcsszó. Ezért ez a változó modulra lokális változó lesz.

Példa:

A `3.c` modulban az **`fgv2`** minden jelző nélkül deklarálásra (2. sor) és definálásra (5-10. sor) is került. Ezért ez a függvény a globális **`fgv2`**.

*A `2.c` modulban is van **`fgv2`** az takarja ezt az **`fgv2`**-t.*

A 4. sorban a **`char txt[]="M text\n;"`** változó deklarációja előtt ott a már látott **`static`** kulcsszó. Ezért ez a változó modulra lokális változó lesz.

A 11-15. sorban definiált **`fgv3`** függvényben szintén deklaráltuk a **`char txt[]="B text\n;"`** változót. Ez a változó egy blokkra lokális változó. Tehát ez csak az **`fgv3`**-ban látszik.

Példa:

A `3.c` modulban az **`fgv2`** minden jelző nélkül deklarálásra (2. sor) és definálásra (5-10. sor) is került. Ezért ez a függvény a globális **`fgv2`**.

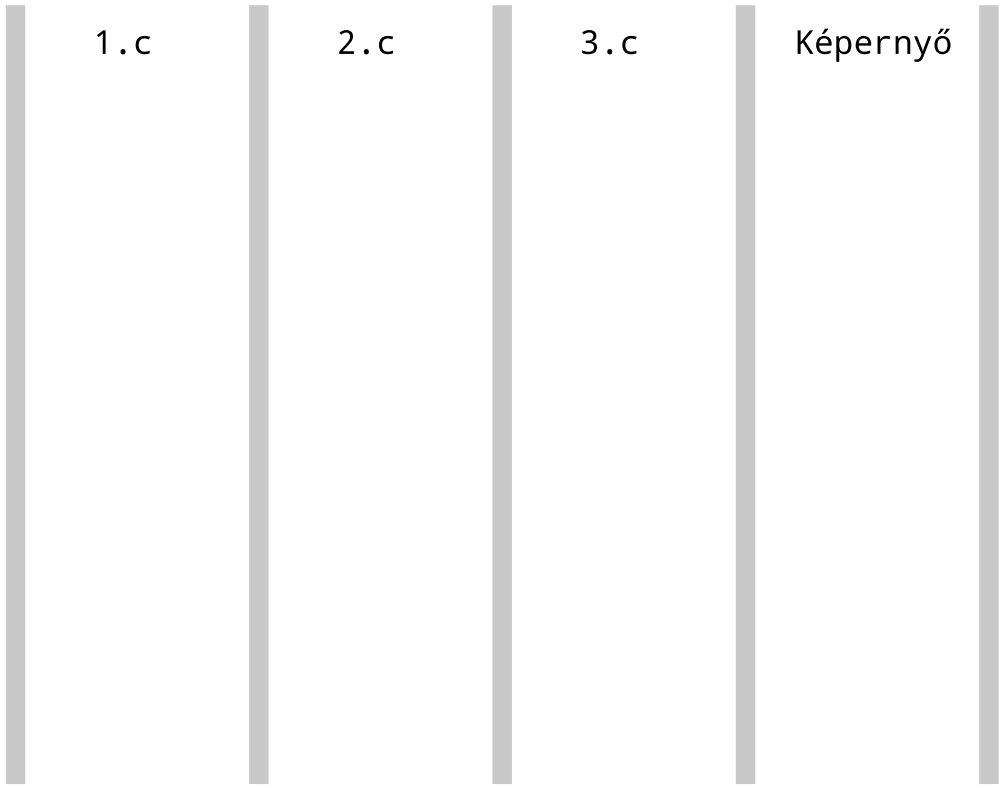
*A `2.c` modulban is van **`fgv2`** az takarja ezt az **`fgv2`**-t.*

A 4. sorban a **`char txt[]="M text\n;"`** változó deklarációja előtt ott a már látott **`static`** kulcsszó. Ezért ez a változó modulra lokális változó lesz.

A 11-15. sorban definiált **`fgv3`** függvényben szintén deklaráltuk a **`char txt[]="B text\n;"`** változót. Ez a változó egy blokkra lokális változó. Tehát ez csak az **`fgv3`**-ban látszik.

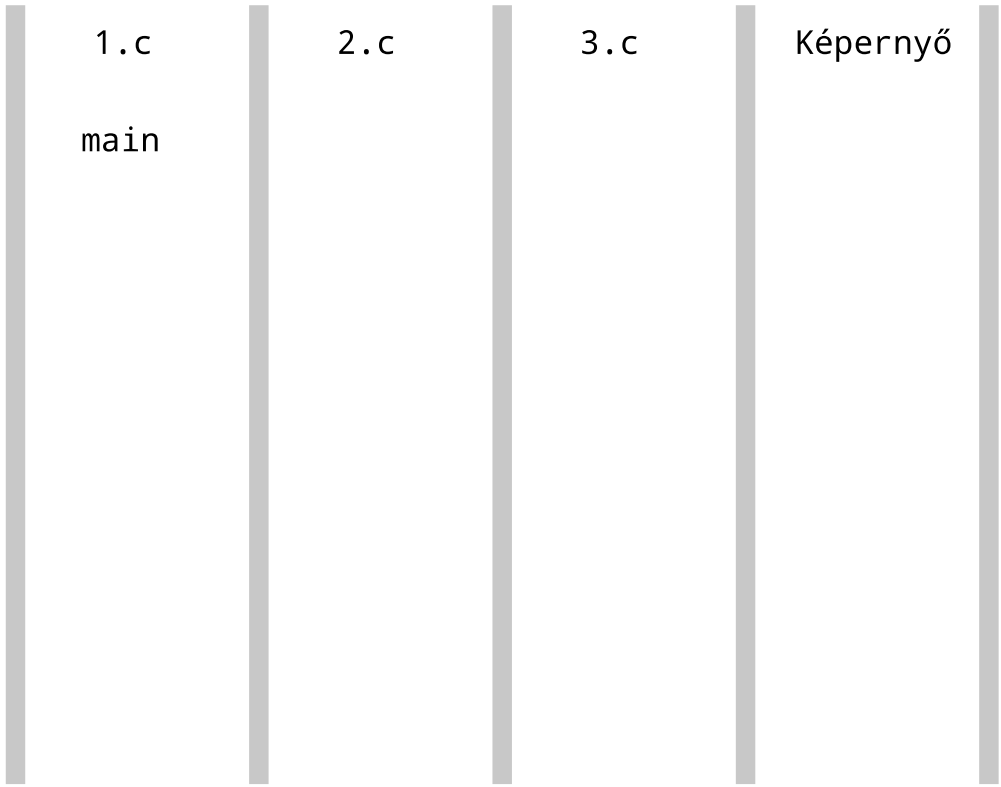
*A **`fgv3`**-ban a **`txt`** változó eltakarja a globális és a modulra lokális, azonos nevű változókat.*

Példa:



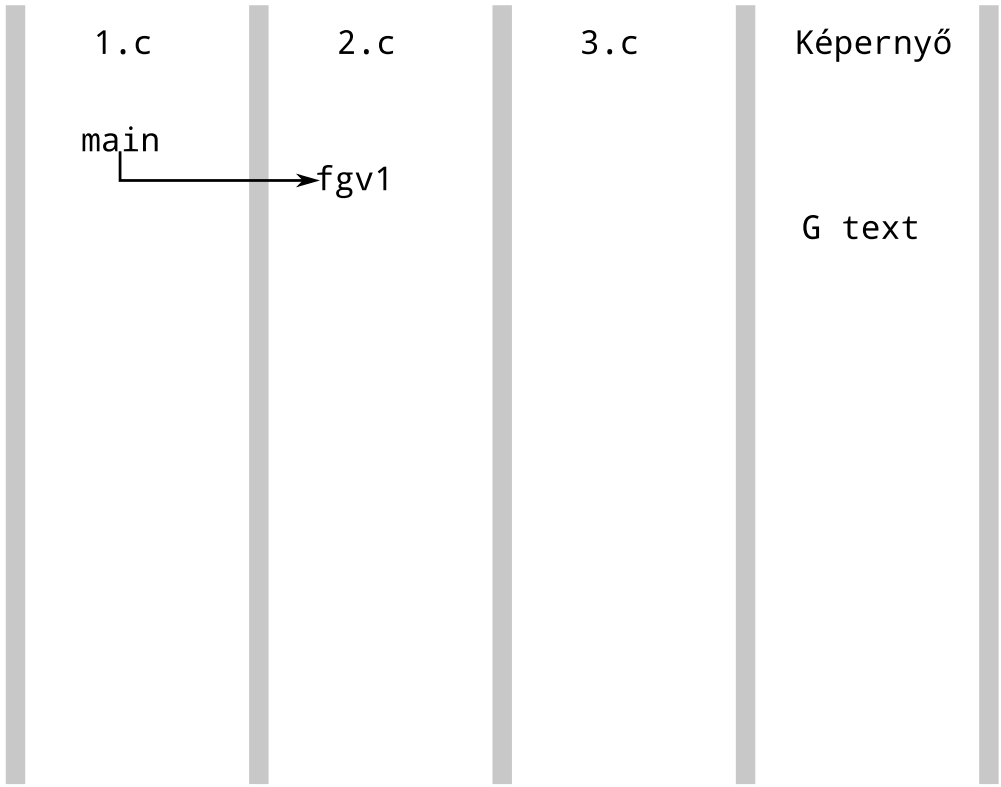
Példa:

Az **`1.c`**-ben a **`main`** elindul.



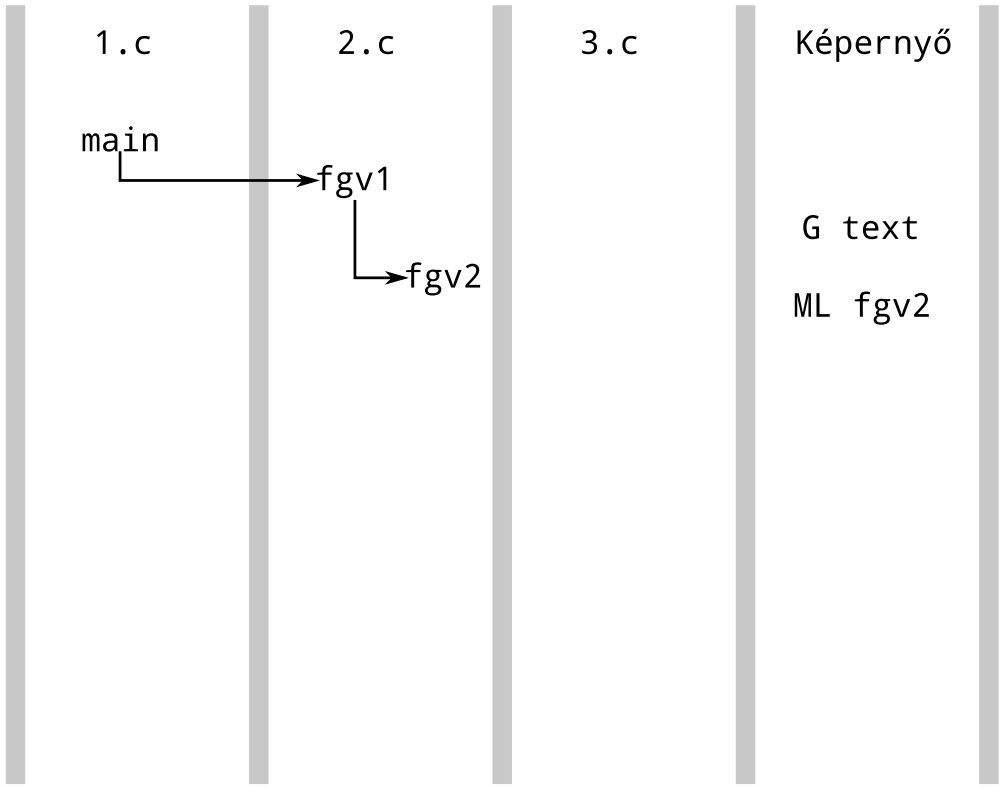
Példa:

Az **`1.c`**-ben a **`main`** elindul.
A **`main`** meghívja a **`2.c`**-ben az **`fgv1`**-et.



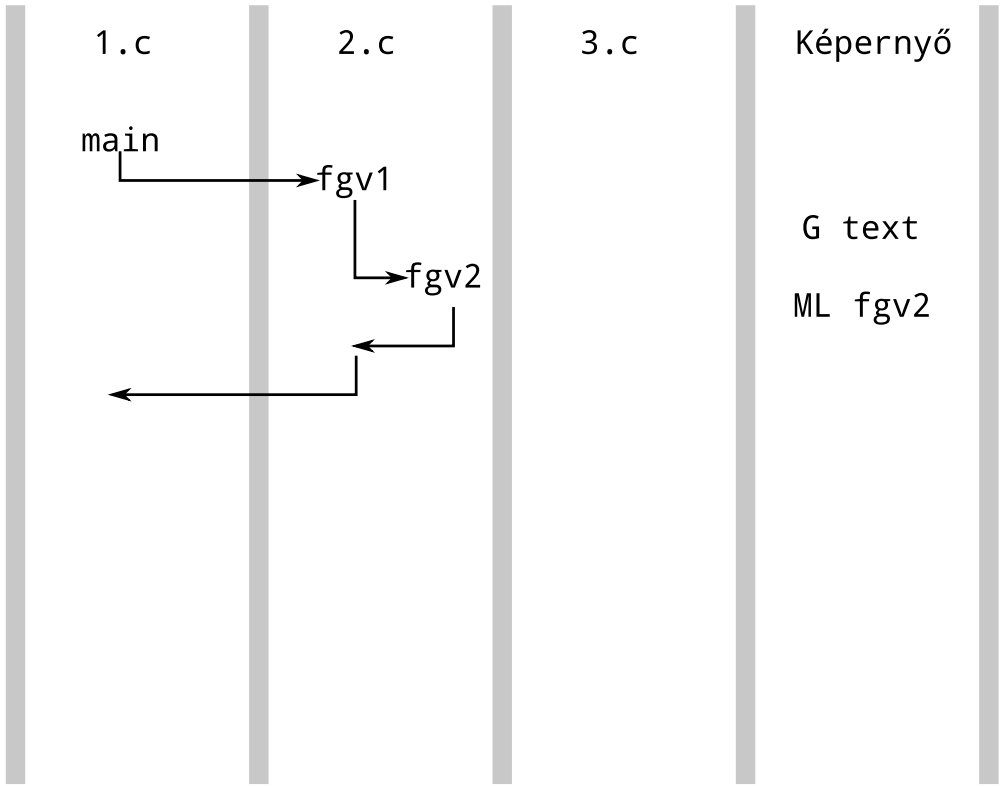
Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.



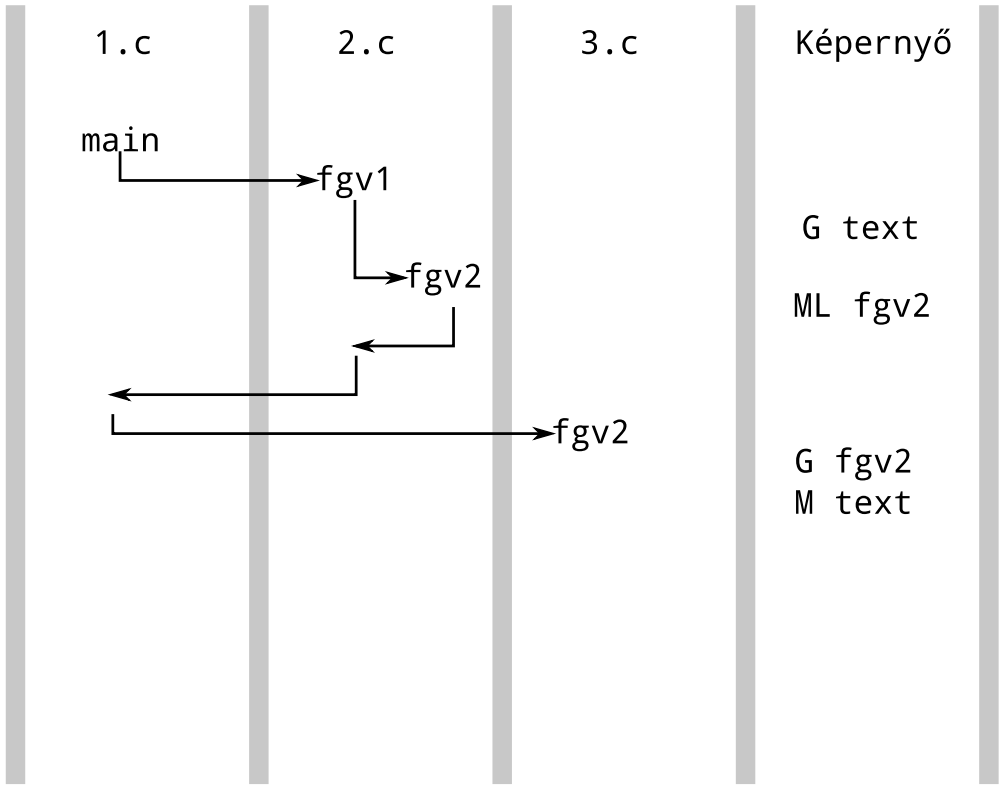
Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.
Az **fgv2** visszatér az **fgv1**-be, majd **fgv1** visszatér a **main**-ba.



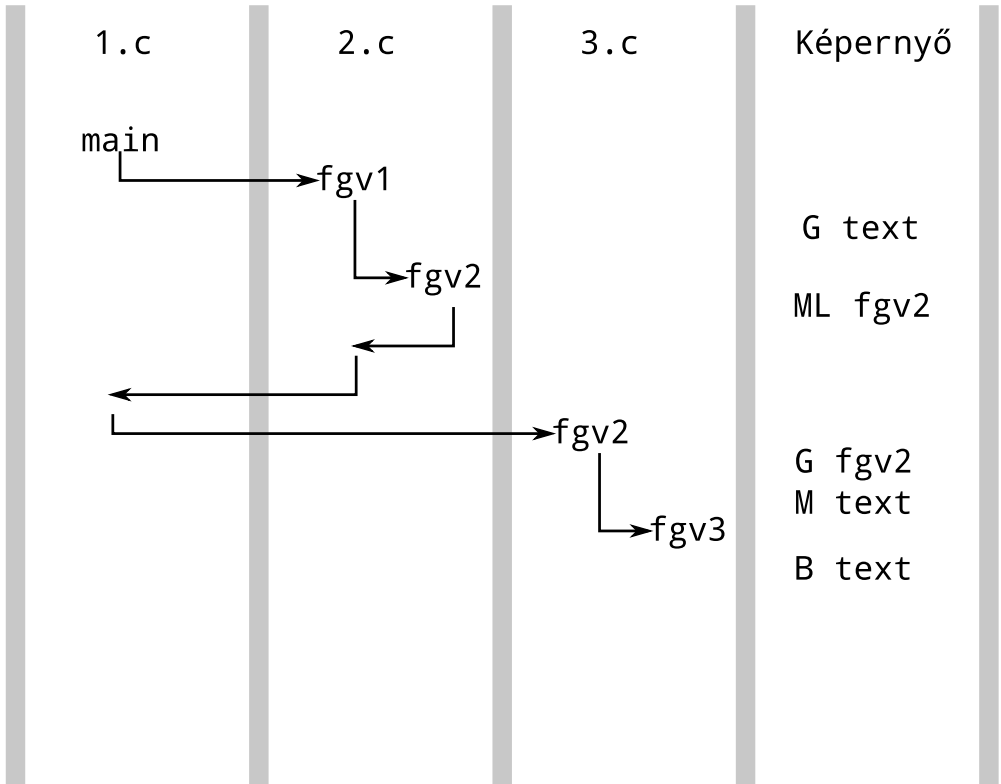
Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.
Az **fgv2** visszatér az **fgv1**-be, majd **fgv1** visszatér a **main**-ba.
A main meghívja a **3.c**-ben lévő **fgv2**-őt.



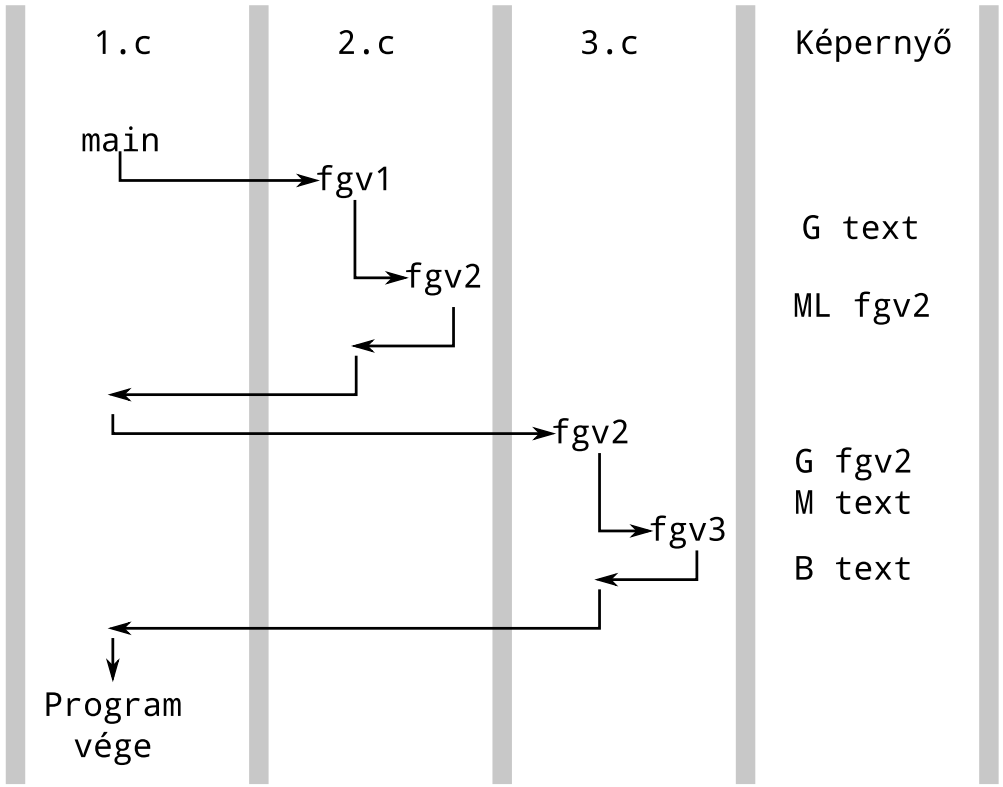
Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.
Az **fgv2** visszatér az **fgv1**-be, majd **fgv1** visszatér a **main**-ba.
A main meghívja a **3.c**-ben lévő **fgv2**-őt.
Az **fgv2** meghívja **3.c**-ben lévő **fgv3**-at.



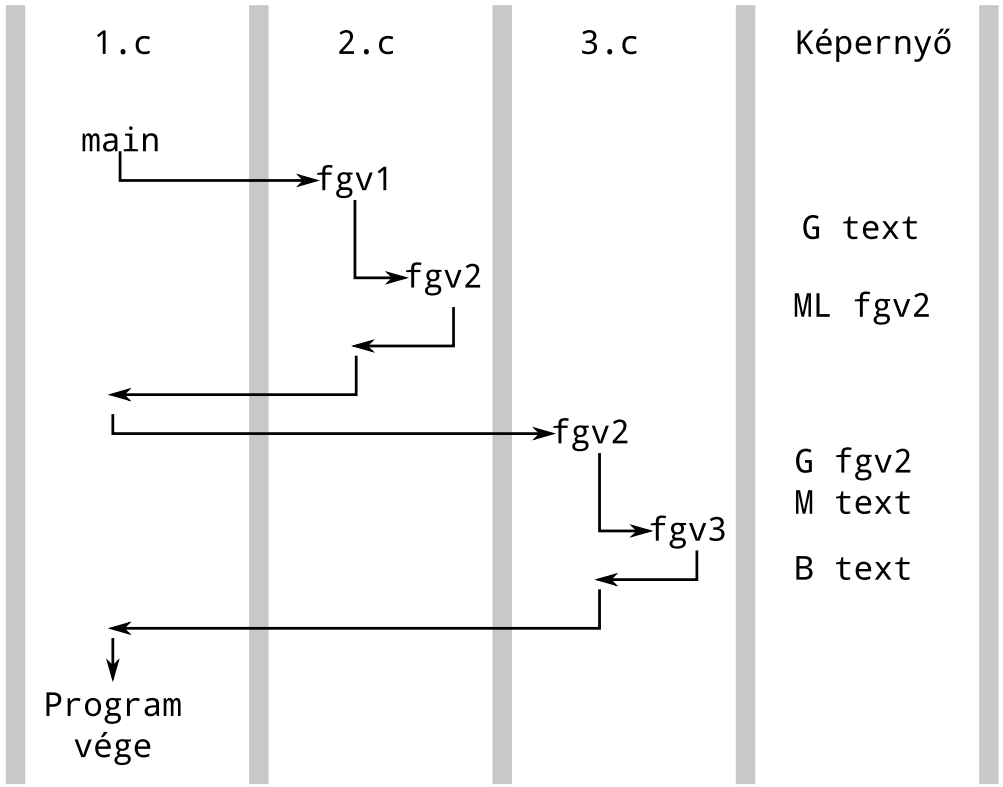
Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.
Az **fgv2** visszatér az **fgv1**-be, majd **fgv1** visszatér a **main**-ba.
A main meghívja a **3.c**-ben lévő **fgv2**-őt.
Az **fgv2** meghívja **3.c**-ben lévő **fgv3**-at.
Az **fgv3** visszatér az **fgv2**-be, majd **fgv2** visszatér a **main**-ba és a program befejeződik.



Példa:

Az **1.c**-ben a **main** elindul.
A **main** meghívja a **2.c**-ben az **fgv1**-et.
Az **fgv1** meghívja a **2.c**-ben lévő **fgv2**-őt.
Az **fgv2** visszatér az **fgv1**-be, majd **fgv1** visszatér a **main**-ba.
A main meghívja a **3.c**-ben lévő **fgv2**-őt.
Az **fgv2** meghívja **3.c**-ben lévő **fgv3**-at.
Az **fgv3** visszatér az **fgv2**-be, majd **fgv2** visszatér a **main**-ba és a program befejeződik.
Pfuhh!



Blokkra lokális változók

A klasszikus blokkra lokális változók dinamikusak.

Tehát futásidőben jönnek létre és futás időben is szünnek meg.

Ennek három következménye van:

1. a változó nem inicializált esetben meghatározatlan értéket tartalmaz,

```
{
int v;
```

2. a változó két függvényhívás között nem létezik, tehát a függvényből kilépéskor tartalmazott értéke is ”megsemmisül”,
3. a deklarációnál végzett inicializálás minden függvénybe lépéskor megtörténik.

```
{
int v=5;
```

Blokkra lokális változók

Egy blokkra lokális változó lehet statikus. Ha statikus, akkor fordítás időben jön létre.

Ennek is három következménye van:

1. a változó, ha külön nem inicializáljuk általában 0 kezdőértéket vesz fel,

```
{
static int v;
```

2. ha a változót deklarációnál inicializáljuk, akkor a program indulásánál felveszi a megadott értéket, de csak akkor,

```
{
static int v=5;
```

3. ha a változó a függvényből kilépésnél egy adott értékkel rendelkezett, akkor az adott függvénybe való következő belépésnél ezzel az értékkel rendelkezik.

Blokkra lokális változók

Megjegyzések:

- Dinamikus változó ezetén az:

```
{                          {
int v=5;                   int v;
                           v=5;
```

  kifejezések azonosak.

- Statikus változó esetén az:

```
{
static int v;
v=5;
```

  minden futáskor természetesen végrehajtásra kerül.

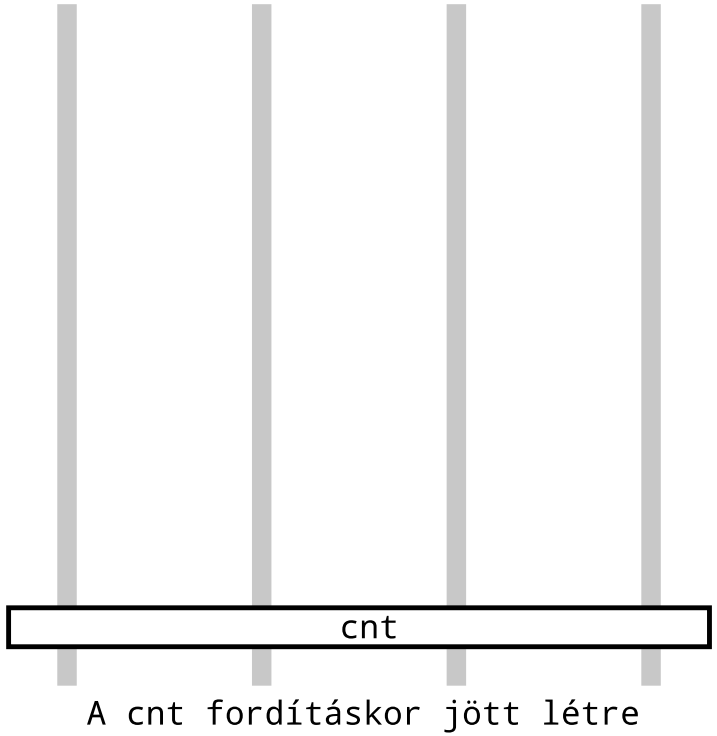
Példa:

Vegyük példának a következő rekurzív függvényt.

```
1.  unsigned fact(unsigned n)
2.    {
3.    static int cnt=0;          // Ez a melyseg szamlalo
4.    unsigned r;                // Seged valtozo
5.    if(n<=1)
6.      {
7.      cnt++;
8.      return 1;
9.      }
10.   cnt++;
11.   r=n*fact(n-1);             // A tenyleges rekurziv hivas
12.   return r;
13.   }
```

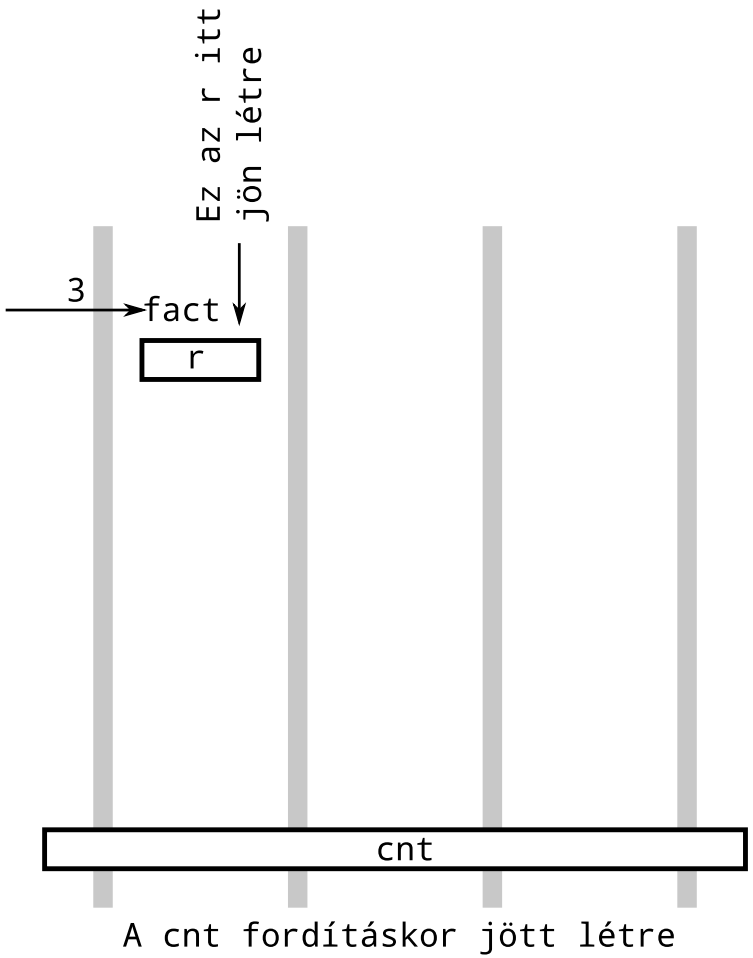
Példa:

A **`cnt`** változó fordítás időben létrejön (3. sor).



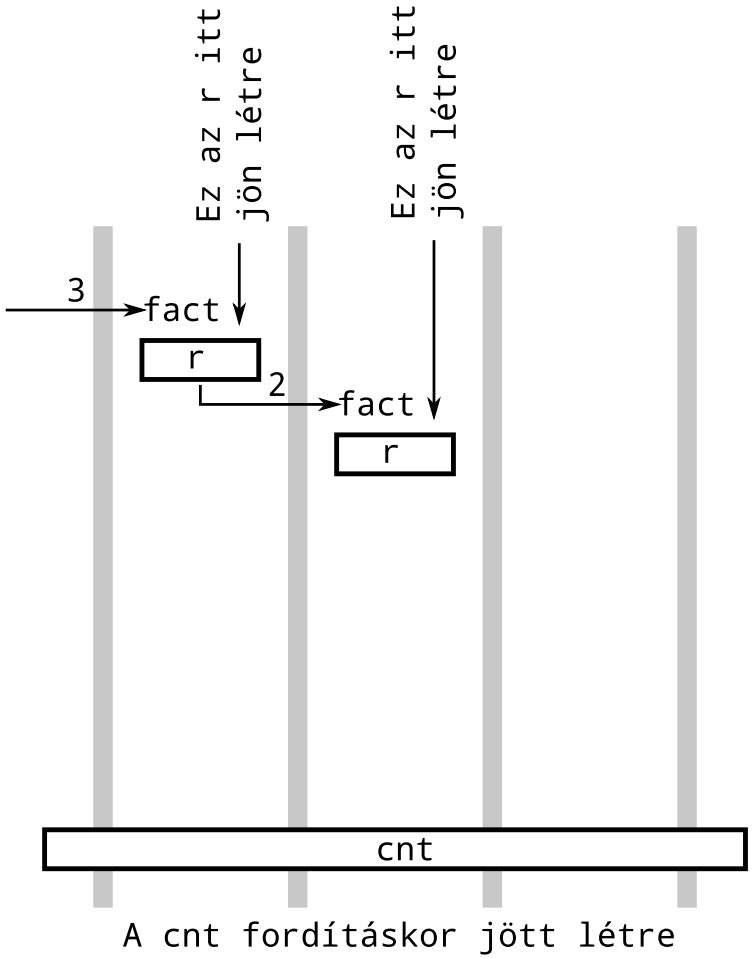
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).



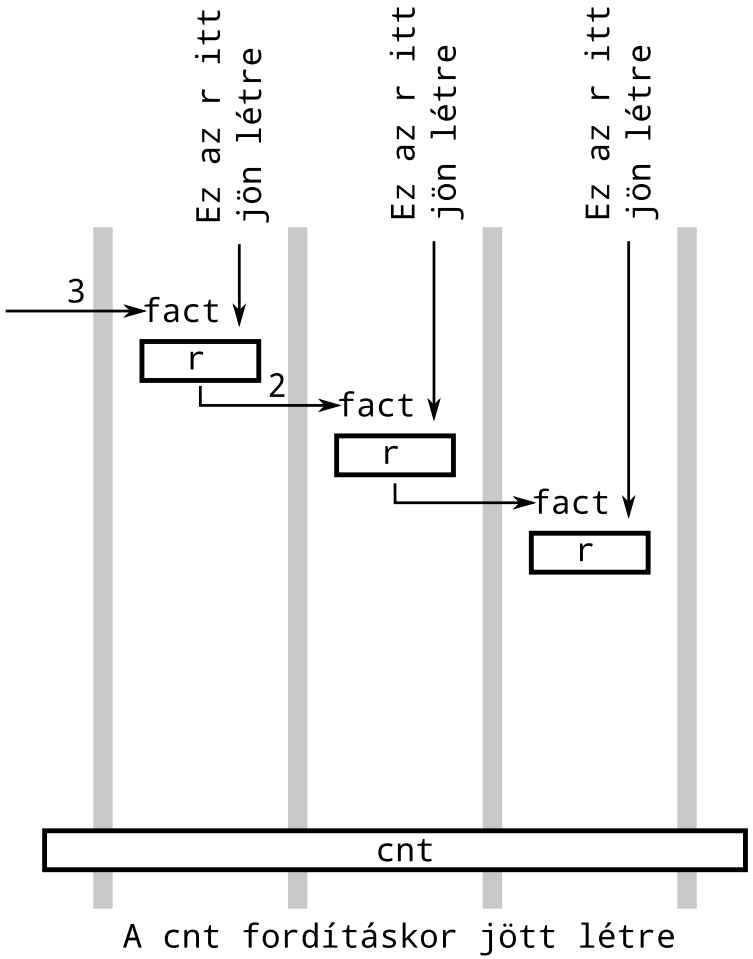
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.



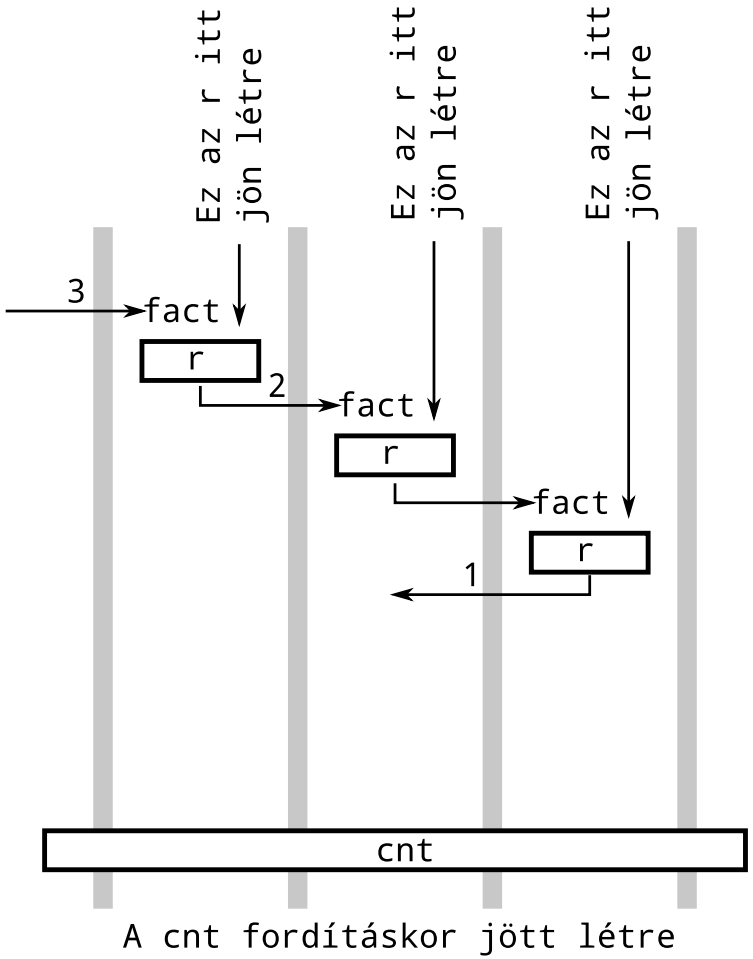
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.
Majd újra meghívja a függvény magát már 1 értékkel.



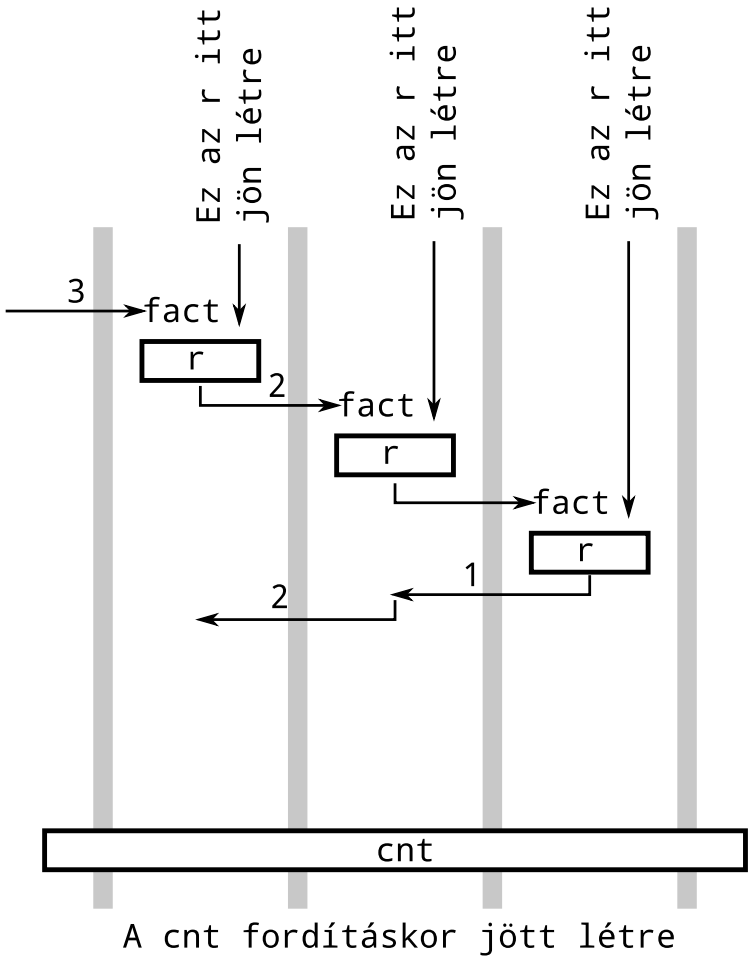
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.
Majd újra meghívja a függvény magát már 1 értékkel.
A függvény visszatér 1 értékkel és az aktuális **r** változó megsemmisül.



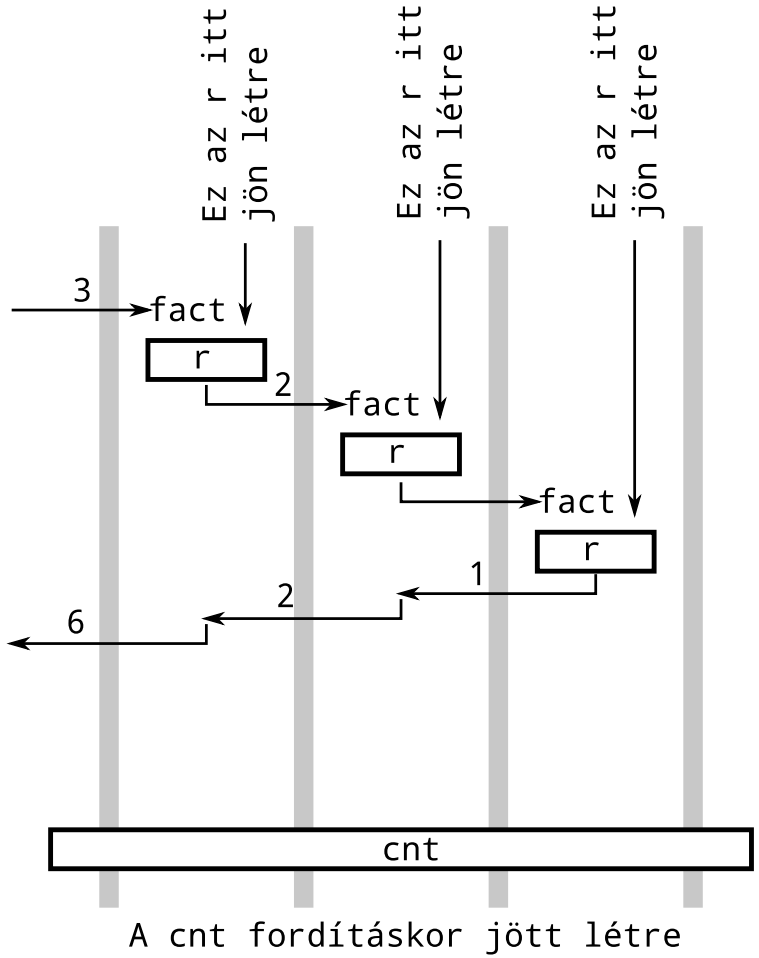
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.
Majd újra meghívja a függvény magát már 1 értékkel.
A függvény visszatér 1 értékkel és az aktuális **r** változó megsemmisül.
A függvény visszatér 1*2 értékkel és az aktuális **r** változó megsemmisül.



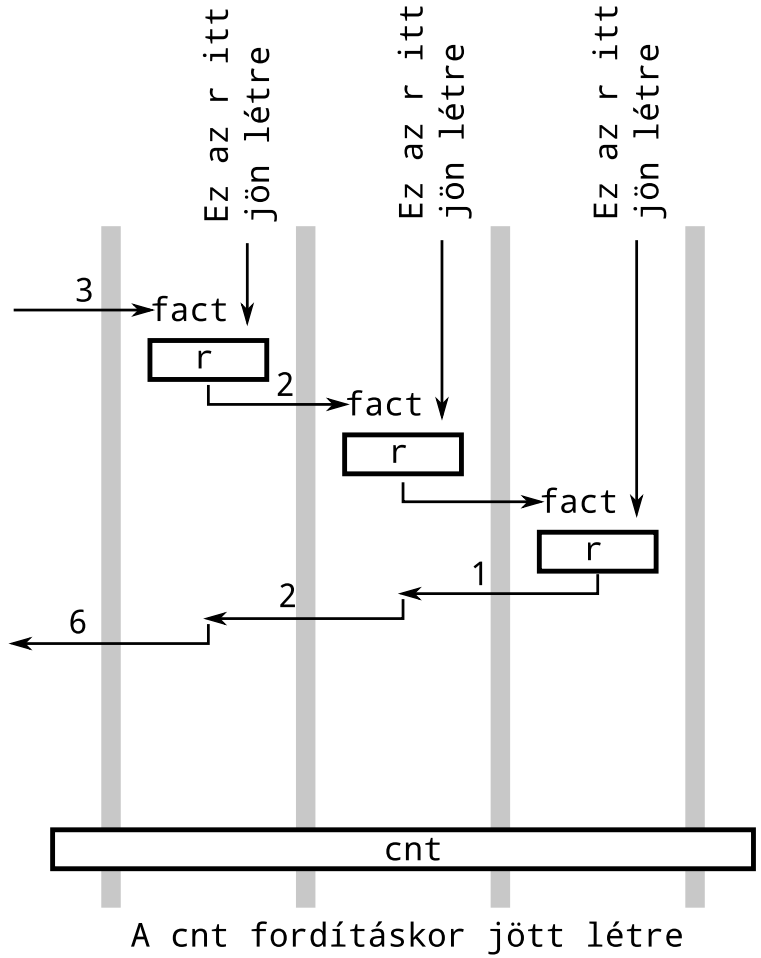
Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.
Majd újra meghívja a függvény magát már 1 értékkel.
A függvény visszatér 1 értékkel és az aktuális **r** változó megsemmisül.
A függvény visszatér 1*2 értékkel és az aktuális **r** változó megsemmisül.
A függvény visszatér (1*2)*3 értékkel és az aktuális **r** változó megsemmisül.



Példa:

A **cnt** változó fordítás időben létrejön (3. sor).
A **fact** függvényt meghívjuk egy 3-as értékkel, ekkor létrejön az aktuális **r** változó (4. sor).
Ha a bemeneti paraméter nagyobb, mint 1, akkor függvény meghívja saját magát (11. sor).
A meghívott **fact** függvényben újra felépül az **r** változó.
Majd újra meghívja a függvény magát már 1 értékkel.
A függvény visszatér 1 értékkel és az aktuális **r** változó megsemmisül.
A függvény visszatér 1*2 értékkel és az aktuális **r** változó megsemmisül.
A függvény visszatér (1*2)*3 értékkel és az aktuális **r** változó megsemmisül.



A **cnt** változó az egész rekurzív eljárás során ugyanaz a változó és hívási mélységet számolja.

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Melyek a C területfoglaló objektumai?

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Melyek a C területfoglaló objektumai?

A változók.

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Melyek a C területfoglaló objektumai?

A változók.

Melyek a C kódgeneráló objektumai?

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Melyek a C területfoglaló objektumai?

A változók.

Melyek a C kódgeneráló objektumai?

A függvények.

Kérdések 1.

Miért alkalmazzák a moduláris programozás módszertant?

a program kódja kezelhető méretű modulokra van bontva, több programozó dolgozhat egyidőben az adott feladaton, egy modult könnyebb tesztelni, mint egy teljes programot, egy már megrt és tesztelt modul többször is felhasználható, ha vannak általános modulok, azok könyvtárrá szerkeszthetők.

Mi a szerepe a linkernek?

Az egyenként lefordított modulokat futtatható kóddá szerkeszti össze

Melyek a C területfoglaló objektumai?

A változók.

Melyek a C kódgeneráló objektumai?

A függvények.

Kérdések 2.

Mit jelent, ha egy változó statikus?

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Ez miért probléma?

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Ez miért probléma?

A dinamikus változó létrejöttekor memória szemetet tartalmaz.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Ez miért probléma?

A dinamikus változó létrejöttekor memória szemetet tartalmaz.

Mit jelent, hogy egy változó globális?

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Ez miért probléma?

A dinamikus változó létrejöttekor memória szemetet tartalmaz.

Mit jelent, hogy egy változó globális?

A változó minden modul függvényéből **láthatóvá tehető**.

Kérdések 2.

Mit jelent, ha egy változó statikus?

A változó területe fordítás időben jön létre.

Mit jelent, ha egy változó dinamikus?

A változó futásidőben jön létre és sznik meg.

Milyen tipikus hibát lehet elközvetni dinamikus változók esetén.

A változót inicializálás nélkül használják fel.

Ez miért probléma?

A dinamikus változó létrejöttekor memória szemetet tartalmaz.

Mit jelent, hogy egy változó globális?

A változó minden modul függvényéből **láthatóvá tehető**.

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Mi a szabály, ha azonos nevű globális, modulra lokális és blokkra lokális változó van a programban?

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Mi a szabály, ha azonos nevű globális, modulra lokális és blokkra lokális változó van a programban?

A globális változót kitakarja a modulra lokális, illetve a blokkra lokális, a modulra lokálist kitakarja a blokkra lokális (ez kerülendő).

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Mi a szabály, ha azonos nevű globális, modulra lokális és blokkra lokális változó van a programban?

A globális változót kitakarja a modulra lokális, illetve a blokkra lokális, a modulra lokálist kitakarja a blokkra lokális (ez kerülendő).

Mit jelent, hogy egy függvény globális?

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Mi a szabály, ha azonos nevű globális, modulra lokális és blokkra lokális változó van a programban?

A globális változót kitakarja a modulra lokális, illetve a blokkra lokális, a modulra lokálist kitakarja a blokkra lokális (ez kerülendő).

Mit jelent, hogy egy függvény globális?

Az adott függvény minden modul függvényéből **hívható**.

Kérdések 3.

Mit jelent, hogy egy változó modulra lokális?

Az adott modul minden függvényéből **láthatóvá tehető**.

Mit jelent, hogy egy változó blokkra lokális?

A változó csak az adott függvényből (blokkból) látható.

Mi a szabály, ha azonos nevű globális, modulra lokális és blokkra lokális változó van a programban?

A globális változót kitakarja a modulra lokális, illetve a blokkra lokális, a modulra lokálist kitakarja a blokkra lokális (ez kerülendő).

Mit jelent, hogy egy függvény globális?

Az adott függvény minden modul függvényéből **hívható**.

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Mit jelent egy blokkra lokális változó deklarációja előtt a `static` kulcsszó?

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Mit jelent egy blokkra lokális változó deklarációja előtt a `static` kulcsszó?

A blokkra lokális változó statikus lesz.

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Mit jelent egy blokkra lokális változó deklarációja előtt a `static` kulcsszó?

A blokkra lokális változó statikus lesz.

Ennek mi a következménye?

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Mit jelent egy blokkra lokális változó deklarációja előtt a `static` kulcsszó?

A blokkra lokális változó statikus lesz.

Ennek mi a következménye?

Ez a változó is csak egyszer, fordításidőben jön létre.

Kérdések 4.

Mit jelent, hogy egy függvény modulra lokális?

A függvény az adott modul függvényeiből hívható.

Mit jelent a `static` kulcsszó egy függvény deklaráció előtt?

A függvény blokkra lokális.

Mit jelent egy blokkra lokális változó deklarációja előtt a `static` kulcsszó?

A blokkra lokális változó statikus lesz.

Ennek mi a következménye?

Ez a változó is csak egyszer, fordításidőben jön létre.

Kérdések 5.

Mi a helyzet dinamikus változó esetén?

Kérdések 5.

Mi a helyzet dinamikus változó esetén?

Az annyiszor jön létre, ahányszor a függvény hívásra kerül, például rekurzív függvények esetén.

Kérdések 5.

Mi a helyzet dinamikus változó esetén?

Az annyiszor jön létre, ahányszor a függvény hívásra kerül, például rekurzív függvények esetén.

Mit jelent a rekurzív függvény fogalom.

Kérdések 5.

Mi a helyzet dinamikus változó esetén?

Az annyiszor jön létre, ahányszor a függvény hívásra kerül, például rekurzív függvények esetén.

Mit jelent a rekurzív függvény fogalom.

A függvény saját magát hívja meg.

Kérdések 5.

Mi a helyzet dinamikus változó esetén?

Az annyiszor jön létre, ahányszor a függvény hívásra kerül, például rekurzív függvények esetén.

Mit jelent a rekurzív függvény fogalom.

A függvény saját magát hívja meg.