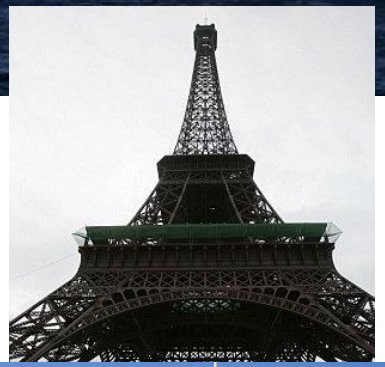
Programozás C nyelven (4. ELŐADÁS)

Sapientia EMTE

2020-21



Hány féle „téglából” építkezünk?



EMLÉKEZTETŐ / KIEGÉSZÍTŐ

Strukturált programozás

I. SZEKVENCIA

- <kifejezés>;(1)
  - értékadás

II.ELÁGAZÁS

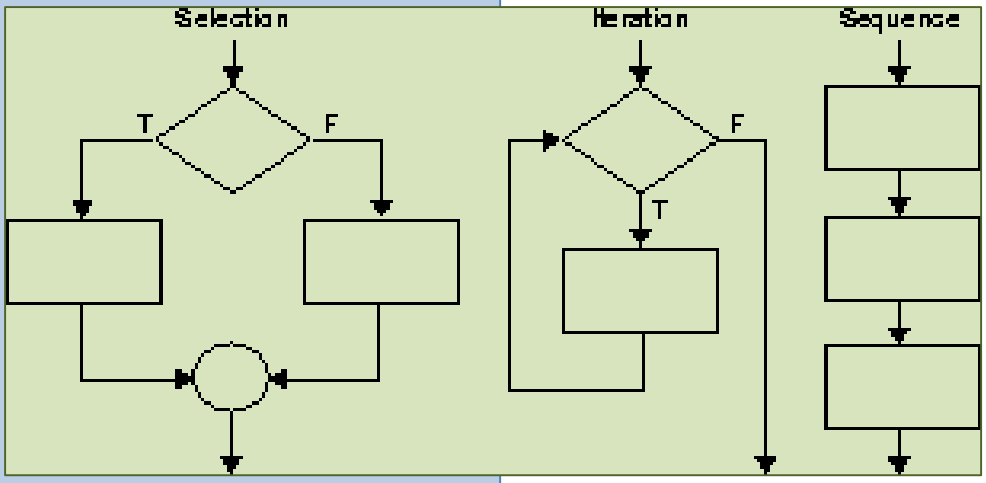
- if-else (2)
- switch (3)

III.CIKLUSOK

- for (4)
- while (5)
- do-while (6)
  - break(7), continue(8), goto(9)

```
if (5 == x);
else {
    a = 1; b = 2;
}
```

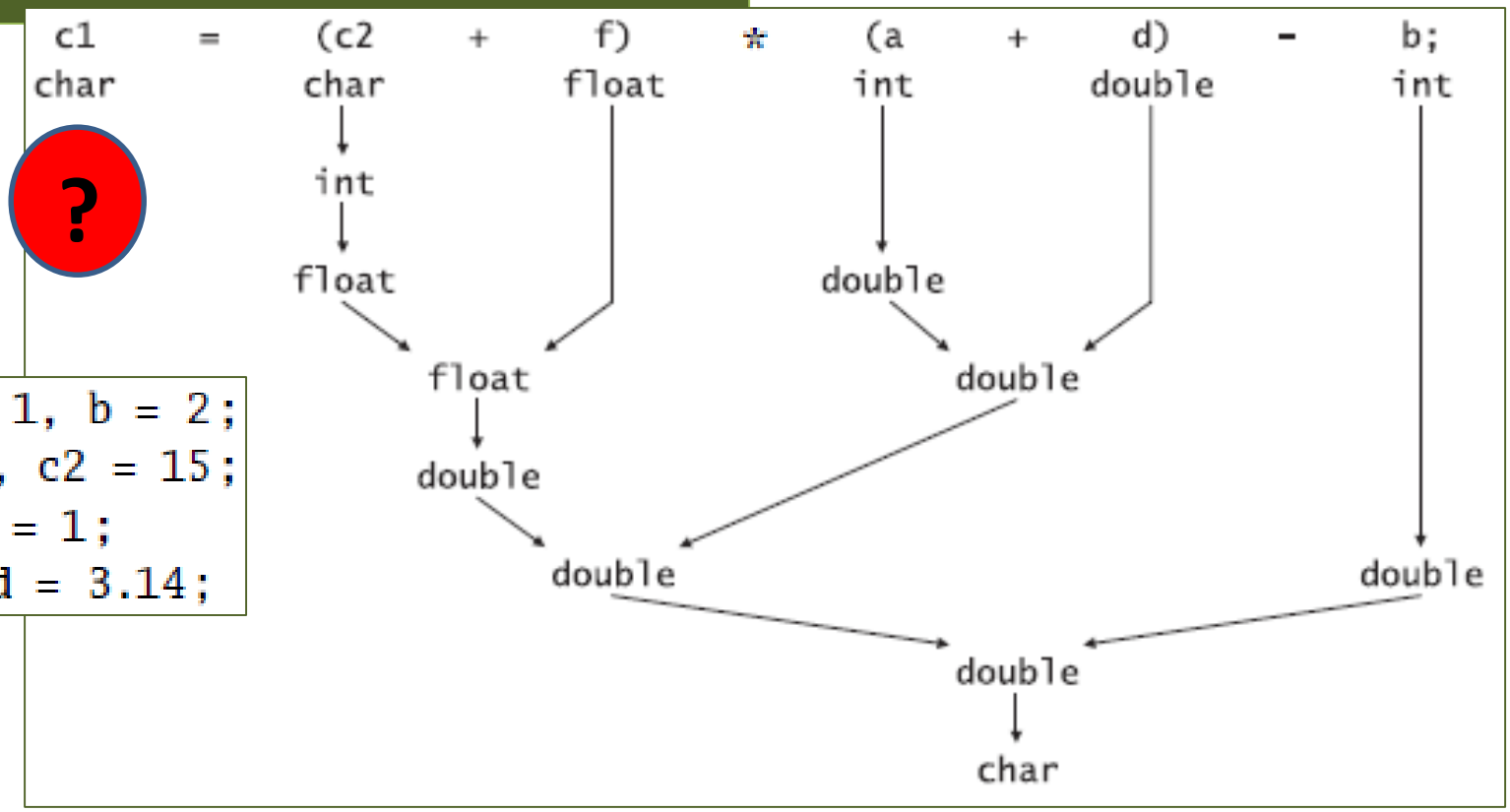
összetett/üres utasítás



? Hány utasításból „építkezünk” ?

KIFEJEZÉSEK

- **operátorok / operandusok**
  - **prioritás-sorrend**
  - **konverziók**



?

```
int a = 1, b = 2;
char c1, c2 = 15;
float f = 1;
double d = 3.14;
```

OPERÁTOROK

Értékadás
<változó> = <kifejezés>

operandusok / operátorok

```
x1 = (-b + sqrt(delta)) / (2*a);
```

```
a = b = c = 0;
```

Jobbról-balra

```
sizeof (<kifejezés>)
sizeof (<típus>)
```

```
sizeof (short)
```

```
short x; double y;
sizeof (x + y)
```

Implicit típuskonverzió

Aritmetikai operátorok
+, -, *, /, %

Összevont operátorok
+=, -=, *=, /=, %=

```
s += x;
```

```
x /= 10;
```

```
int i = 1, j = 1, x, y;
x = ++i; y = j--;
```

?

OPERÁTOROK

Összehasonlítási operátorok

`==, !=, <, <=, >, >=`

Logikai operátorok

`!, &&, ||`

```
while( i < n && j < m ){…}
```

CAST operátor

`(<tipus>)<operandus>`

```
double x = 3.14;
printf("%i",(int)x);
```

?

```
int x = 3;
printf("%f",(float)x / 2);
```

Feltételes operátor

`< > ? < > : < >`

```
max = a > b ? a : b;
```

Vessző operátor

$<kif_1>,<kif_2>,...,<kif_n>$

```
for ( i=1,j=n ; i<j ; ++i,--j ){…}
```

OPERANDUSOK

- **Változók**
- **Konstansok**
- **Függvényhívások**

```
x1 = (-b + sqrt(delta)) / (2*a);
```

0, 7, 19, 25432 (10-es számrendszerben)
0, 013, 0257 (8-as számrendszerben)
0x1a ,0x234, 0XAB2F (16-os számrendszerben)

65 ⟺ 0101 ⟺ 0X41

```
1   - int típusú 1-es
1U  - unsigned int típusú 1-es
1L  - long int típusú 1-es
1UL - unsigned long int típusú 1-es
```

3.14, -17.65, 1., 1.0, 0.1, .1, -0.025e-1,-12.6E2

-1.F 0.01E+5L

'A', 'a', '0', '9', '!', '+', ' '

'\ooo' '\xhh'

'\a'	BEL	csengő
'\b'	BS	visszalépés
'\f'	FF	lapdobás
'\n'	LF	új sor (soremelés)
'\r'	CR	kocsi vissza
'\t'	HT / TAB	vízszintes tabulátor
'\v'	VT	függőleges tabulátor
'\\'	\	backslash
'\''	'	aposztróf
'\"'	"	idézőjel
'\?'	?	kérdőjel

"Alma"

"két sorba tört hosszú\
 karakterlánc konstans"

"a kettő" "egy lesz"

```
#define PI 3.14
```

INT_MAX

```
#include <limits.h>
```

Találós kérdés (1)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

Találós kérdés (2)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

Október 31

December 25

Találós kérdés (3)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

OKT
31

DEC
25

Találós kérdés (4)

- Mi egy közös vonás a halloween-ben és a karácsonyban?

OKT

31

DEC

25

```
25 : 8 = 3 : 8 = 0
24       0
--       -
=1       3
```

Függvények

```c
#include <stdio.h>
#include <math.h>
int main(){
    double x;
    scanf("%ld", &x);
    printf("Gyoke: %ld\n", sqrt(x));
    return 0;
}
```

Könyvtári függvények

```c
#include <iostream>
int main(){
    int a, b;
    scanf("%i%i", &a, &b);
    printf("LNKO: %i\n", lnko(a,b));
    return 0;
}
```

SAJÁT függvények

`main` függvény

visszatérített eredmény típusa

név / azonosító

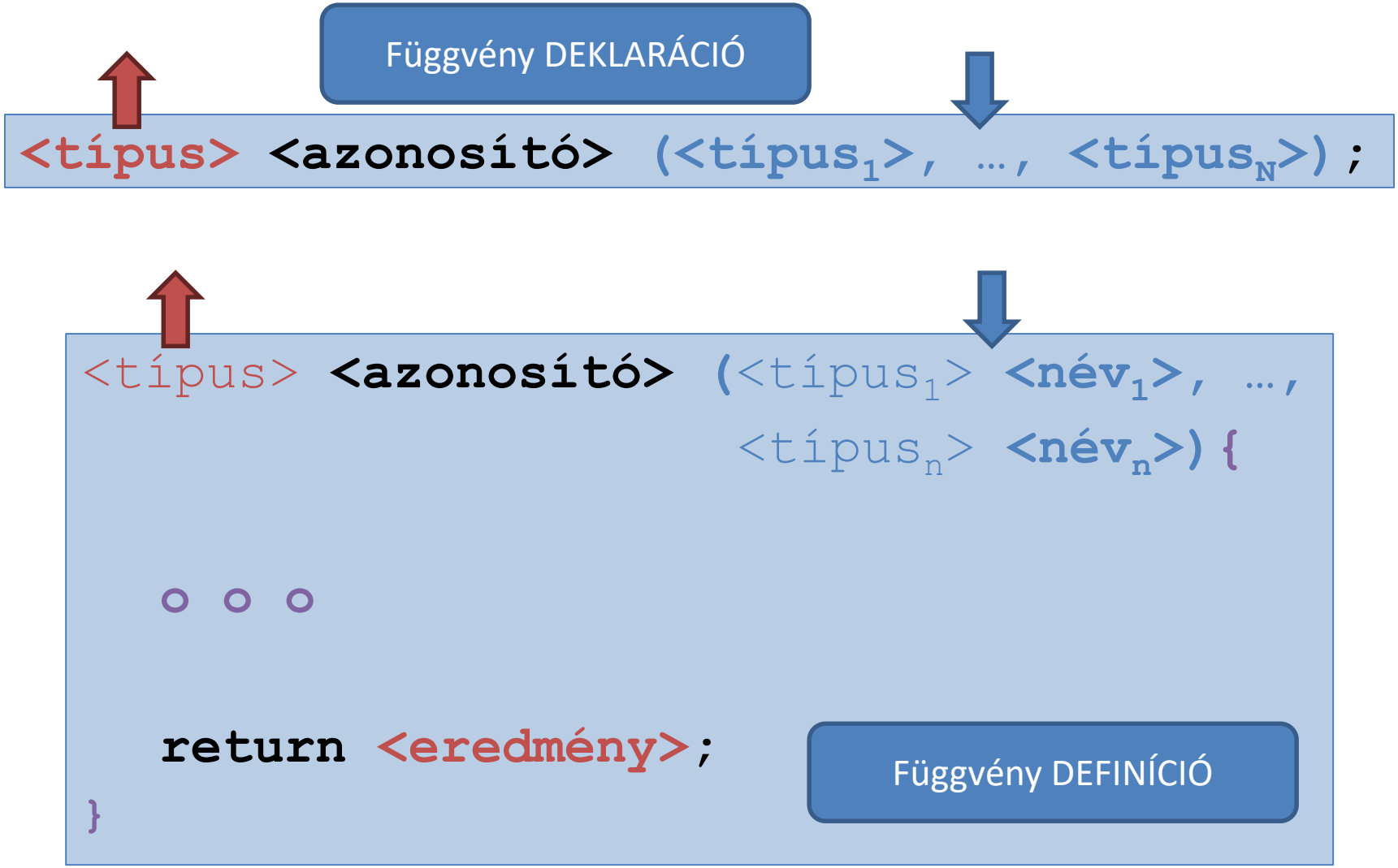
paraméter-lista <típus> <azonosító>

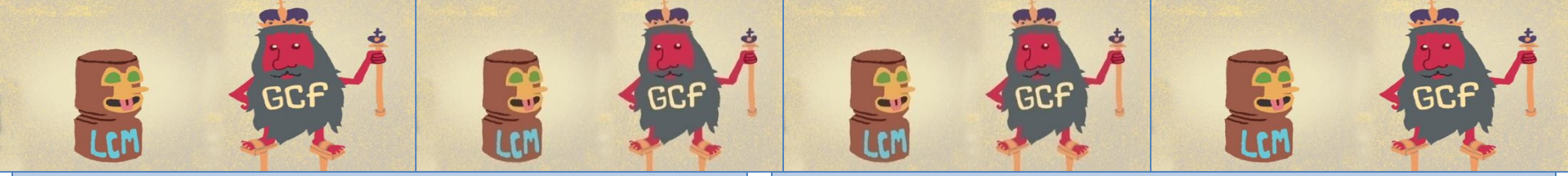
```
int main(){
   . . .
  return 0;
}
```

függvény törzse / magva

Az operációs rendszer **hívja meg**, és ennek téríti vissza a 0 hibakódot

Saját (valódi)függvények





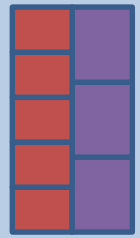

```c
#include <stdio.h>
int lnko(int,int);
int main(){
    int a, b, c;
    scanf("%i%i",&a,&b);
    c = lnko(a,b);
    printf("%i", c);
    return 0;
}
int lnko(int x, int y){
    while (x != y){
     if (x < y) { y -= x; }
     else { x -= y; }
    }
    return x;
}
```

deklaráció

függvény hívás

definíció

```c
#include <stdio.h>
int lkkt(int,int);
int main(){
    int a, b;
    scanf("%i%i",&a,&b);
    printf("%i", lkkt(a,b));
    return 0;
}
int lkkt(int x, int y){
    int sx = x, sy = y;
    while (sx != sy){
      if (sx < sy) {sx += x;}
      else {sy += y;}
    }
    return sx;
}
```



lnko - lépésenként


```
#include <stdio.h>
int lnko(int,int);
int main(){
    int a, b, c;
    scanf("%i%i",&a,&b);
    c = lnko(a,b);
    printf("%i", c);
    return 0;
}
int lnko(int x, int y){
    while (x != y){
     if (x < y) {y -= x;}
     else { x -= y; }
    }
    return x;
}
```

12 21_

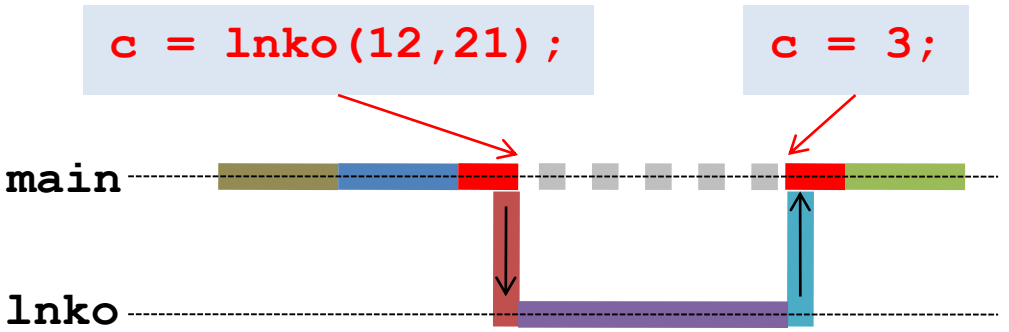
12 21
3_

színészek

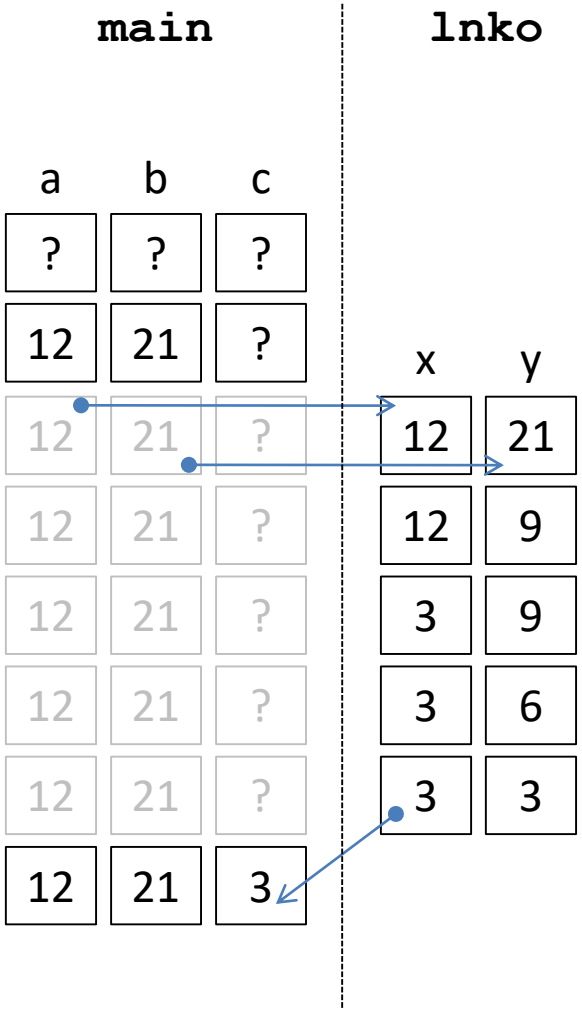
effectív/aktuális paraméterek

formális paraméterek

szereplők/ dublőrök



lnko - lépésenként




```
#include <stdio.h>
int lnko(int,int);
int main(){
    int a, b, c;
    scanf("%i%i",&a,&b);
    c = lnko(a,b);
    printf("%i", c);
    return 0;
}
int lnko(int x, int y){
    while (x != y){
     if (x < y) { y -= x; }
     else { x -= y; }
    }
    return x;
}
```

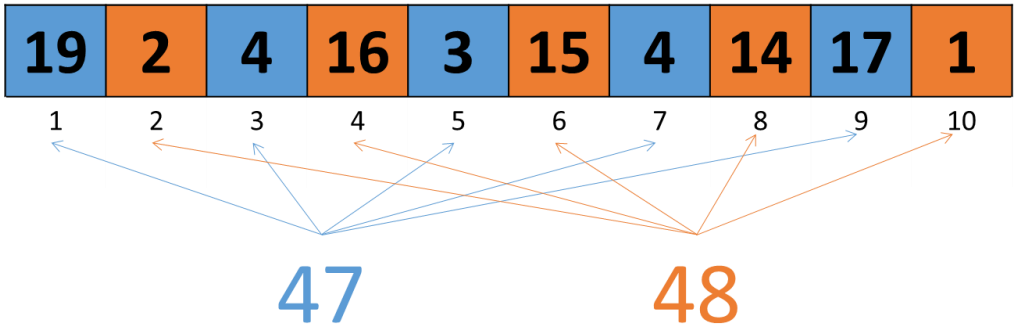
```
12 21_
```

```
12 21
3_
```

*Beolvas n;
*Generál a[0..2*n-1];
*Meghatároz nyerő stratégia (páros/páratlan);
bal ← 0; jobb ← 2*n-1;
minden i=1,n **végezd**
    ha (bal a nyerő) **akkor**
        *PC lép bal
    különben
        *PC lép jobb
    vége ha
    *USER választ bal/jobb
    ha (bal-t választott) **akkor**
        *USER lép bal
    különben
        *USER lép jobb
    vége ha
vége minden
*Kiír pontszámok;

FÜGGVÉNYEK

`kiir_allas(…);`



ÖSSZEFOGLALÓ

- **(valódi) FÜGGVÉNYEK**
  - deklaráció / definíció
  - formális/effektív paraméterek
  - paraméter-átadás / `return` (STACK)
  - hívó/hívott függvény

- **Lnko / Lkkt számítás technikája**