



# Algoritmusok és adatstruktúrák 2. előadás

Pusztai Pál  
[pusztai@sze.hu](mailto:pusztai@sze.hu)

# Tartalom

- Algoritmusok megadása
- Egy mintafeladat – a legnagyobb közös osztó
  - Algoritmus-megadási módszerek
- Strukturált algoritmusok tervezése
  - A szekvencia, a szelekció és az iteráció vezérlőszerkezetek
  - A struktúradiagram- és pseudokód-jelölés
- Kidolgozott feladatok
  - Kör terület, kerület
  - Másodfokú egyenlet megoldása
  - Legkisebb osztó meghatározás
  - Faktoriális számítás
  - Karakterek feldolgozása



## Algoritmusok megadása

- Szöveges leírás
  - Az algoritmust szavakkal, mondatokkal írjuk le.
- Folyamatábra
  - Az algoritmus folyamatát kifejező jelölésmód, amelyen jól követhető az algoritmus lépéseinek sorrendje, a megoldás menete.
- D-diagram
  - Olyan strukturált folyamatábra, amely csak a szekvencia, szelekció, iteráció vezérlőszerkezeteket engedi meg, csak ezekből építkezik.
- Struktúradiagram
  - Az algoritmusok felülről-lefelé történő tervezését támogató eszköz.
- Struktogram
  - Az algoritmusok vezérlőszerkezeteit „dobozokkal” kifejező módszer.
- Pszeudokód
  - Olyan „utasításszintű” leíró eszköz, amelynek vezérlőszerkezeteit a magas szintű programozási nyelvekben „megszokott” módon, vagy ahhoz nagyon hasonlóan írjuk le.

## Algoritmusok megadása

- Megjegyzés:
  - Bizonyos feladatok megoldásánál feltesszük, hogy az input adatok már rendelkezésre állnak, ezeket megkapjuk, míg az output eredményeket visszaadjuk.
  - Ilyenkor a megoldást egy **szubrutin** végzi, a megoldás csak az érdemi részt tartalmazza, az esetleges adatbekérés és eredménykiírás, a szubrutint hívó programrész (pl. a főprogram) feladata.

## Mintafeladat

- **Feladat:** Határozzuk meg két egész szám legnagyobb közös osztóját!



## Mintafeladat

- **Feladat:** Határozzuk meg két egész szám legnagyobb közös osztóját!

- **Megoldás:** Euklidesz (kb. 2300 évvel ezelőtt):

Az  $a$  és  $b \neq 0$  számokra végezzük el a maradékos osztást, azaz

$$a = bq_1 + r_1 \quad (0 \leq |r_1| < |b|).$$

Ezután  $b$  és  $r_1$  között is,

$$b = r_1q_2 + r_2 \quad (0 \leq |r_2| < |r_1|)$$

és így tovább. A maradékok  $(r_1, r_2, \dots)$  abszolút értékben monoton csökkennek, ezért bekövetkezik az

$$r_{n-1} = r_nq_{n+1} + r_{n+1} \quad (0 < |r_{n+1}| < |r_n|)$$

$$r_n = r_{n+1}q_{n+2}$$

állapot, azaz a maradék  $(r_{n+2})$  nulla lesz. Az utolsó nem nulla maradék  $(r_{n+1})$  az  $a$  és  $b$  számok abszolút értékben legnagyobb közös osztóját adja.



## Mintafeladat

- Szöveges leírás:
  1. Ha  $b$  értéke 0, akkor  $a$  értéke a megoldás és készen vagyunk. (Először ezt vizsgáljuk, hogy a nullával való osztást elkerüljük.)
  2. Ha  $b$  értéke nem 0, akkor osszuk el  $a$  értékét  $b$  értékével maradékosan és jelölje az osztás maradékát  $r$ .
  3. Legyen  $a$  új értéke  $b$ ,  $b$  új értéke  $r$  és folytassuk a végrehajtást az 1. lépéssel!

	$a$	$b$	$r$	$a$	$b$	$r$	$a$	$b$	$r$	$a$	$b$	$r$	$a$	$b$	$r$
1.	40	18	4	6	9	6	-10	-8	-2	0	21	0	0	0	0
2.	18	4	2	9	6	3	-8	-2	0	21	0				
3.	4	2	0	6	3	0	-2	0							
4.	2	0		3	0										

**Értékek alakulása osztással**

## Mintafeladat

- Kivonáson alapuló algoritmus:
  1. Ha  $a$  értéke megegyezik  $b$  értékével, akkor ez az érték a megoldás és készen vagyunk.
  2. Ha  $a$  értéke nem egyezik meg  $b$  értékével, akkor a nagyobbik számból vonjuk ki a kisebbet és folytassuk a végrehajtást az 1. lépéssel!
- Megjegyzés
  - Az algoritmus csak  $a > 0$  és  $b > 0$  egészekre működik.

	$a$	$b$
1.	6	9
2.	6	3
3.	3	3

**Értékek alakulása kivonással**





## Mintafeladat

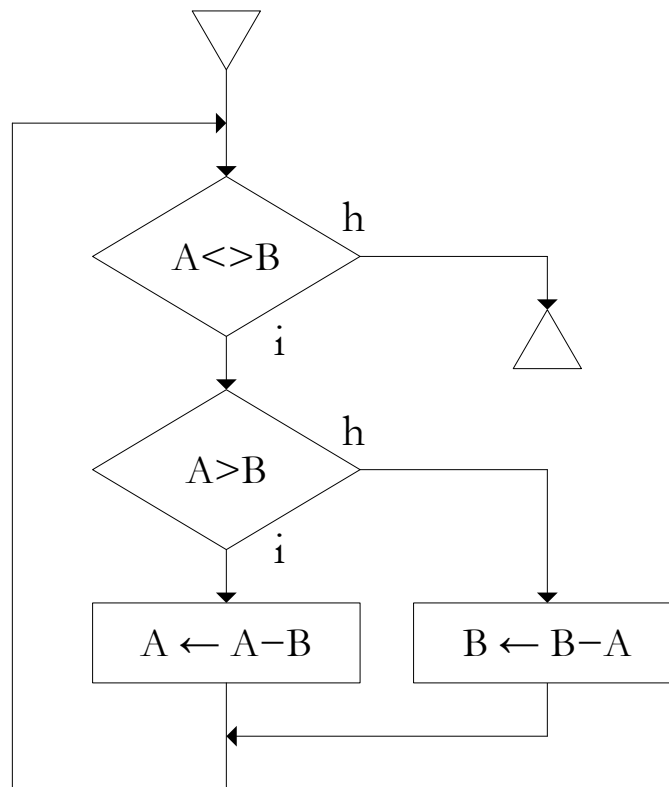
- **Feladat:** Készítsük el a kivonáson alapuló algoritmust az egyes algoritmus megadási módszerekkel!

Funkció	Azonosító	Típus	Jelleg
Az egyik szám	A	Egész	I, M, O
A másik szám	B	Egész	I, M
A rekurzív függvény eredménye	ER	Egész	O

- **Megjegyzés:**
  - Az A és B változók munka jellegét az indokolja, hogy értékük megváltozik az algoritmus során, A output jellegét pedig az, hogy itt „keletkezik” az eredmény.
  - Az ER változót csak a rekurzív megoldásban, a strukturáltság megőrzése érdekében használtuk.

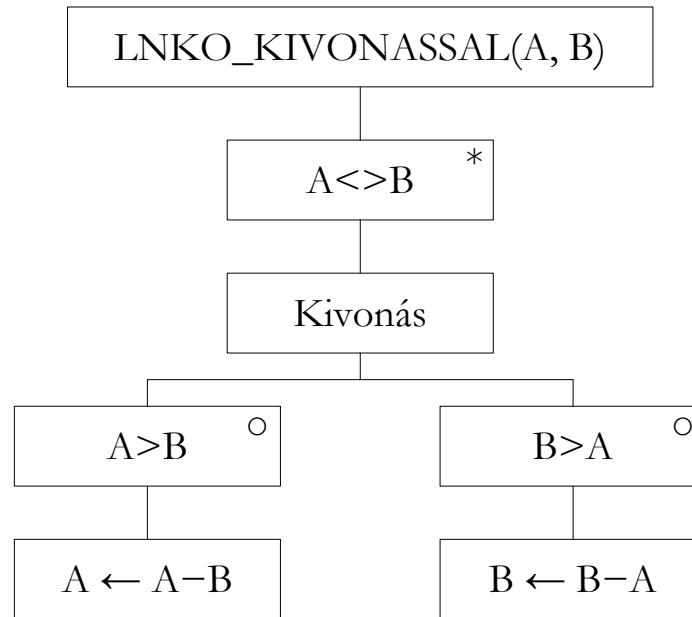
# Mintafeladat

LNKO\_KIVONASSAL(A, B)



Folyamatábra

# Mintafeladat



**Struktúradiagram**

## Mintafeladat

LNKO\_KIVONASSAL(A, B)

A<>B	
//	A>B //
igaz	hamis
$A \leftarrow A - B$	$B \leftarrow B - A$

**Struktogram**

## Mintafeladat

- Pszeudokód

LNKO\_KIVONASSAL(A,B)

**while**  $A \neq B$

**if**  $A > B$

$A \leftarrow A - B$

**else**

$B \leftarrow B - A$

/\* Rekurzív függvénnnyel \*/

LNKO\_KIVONASSAL(A,B)

**if**  $A = B$

$ER \leftarrow A$

**else**

**if**  $A > B$

$ER \leftarrow \text{LNKO\_KIVONASSAL}(A - B, B)$

**else**

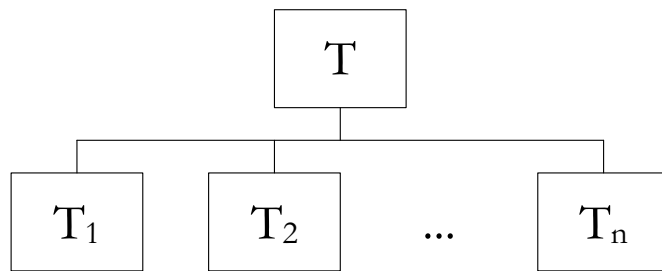
$ER \leftarrow \text{LNKO\_KIVONASSAL}(A, B - A)$

**return** ER



## Strukturált algoritmusok tervezése

- Szekvencia:
  - Olyan vezérlőszerkezet, amelynek során az egyes résztevékenységeket sorban, egymás után hajtjuk végre.
- Jelentés:
  - A  $T$  tevékenység végrehajtása a  $T_1, T_2, \dots, T_n$  résztevékenységek egymás utáni végrehajtásából áll.



Strukturadiagram-jelölés

$T_1$   
 $T_2$   
 $\dots$   
 $T_n$

Pszekdokód-jelölés

## Strukturált algoritmusok tervezése

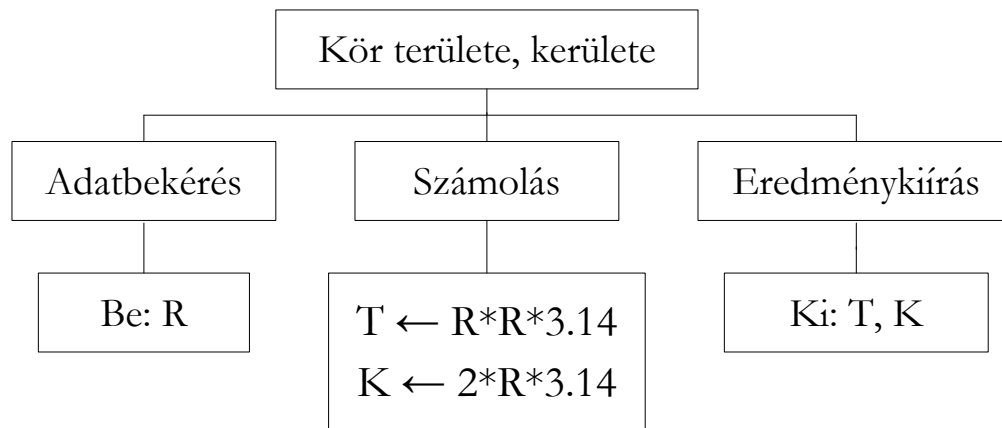
- **Feladat:** Határozzuk meg egy adott sugarú kör területét és kerületét!



## Strukturált algoritmusok tervezése

- **Feladat:** Határozzuk meg egy adott sugarú kör területét és kerületét!

Funkció	Azonosító	Típus	Jelleg
A kör sugara	R	Valós	I
A kör területe	T	Valós	O
A kör kerülete	K	Valós	O



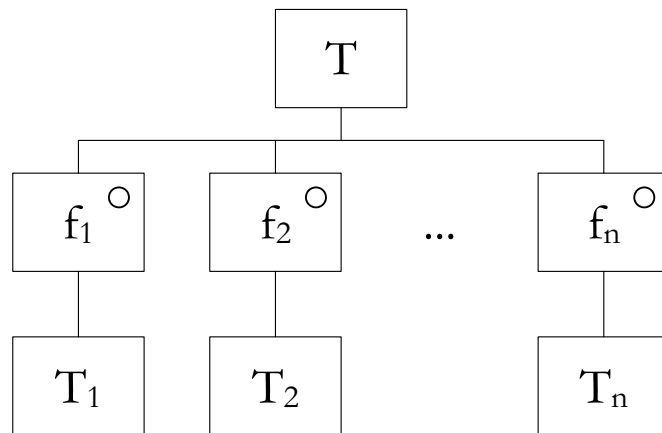
```

/* Kör területe, kerülete */
Be: R
T ← R*R*3.14
K ← 2*R*3.14
Ki: T,K
    
```



# Strukturált algoritmusok tervezése

- Szelekció:
  - Olyan vezérlőszerkezet, amellyel kiválasztható a végrehajtandó résztevékenység.
- Jelentés:
  - A  $T$  tevékenység végrehajtása annak a  $T_i$  résztevékenységnek a végrehajtását jelenti, amelyhez tartozó  $f_i$  feltétel igaz. A feltételek egymást páronként kizárják.



Strukturadiagram-jelölés

```

if f1      T1
else if f2  T2
...
else if fn  Tn
    
```

Pszudokód-jelölés

## Strukturált algoritmusok tervezése

- **Feladat:** Oldjuk meg az  $ax^2+bx+c=0$  másodfokú egyenletet, ahol az  $a, b, c$  együtthatók valós számok!

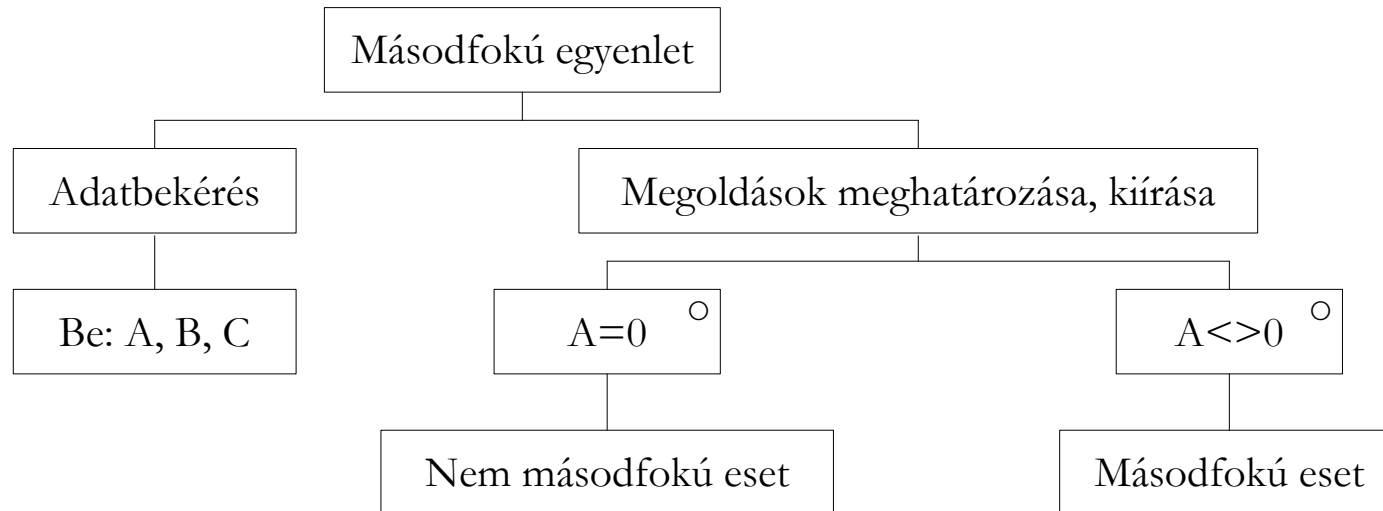


## Strukturált algoritmusok tervezése

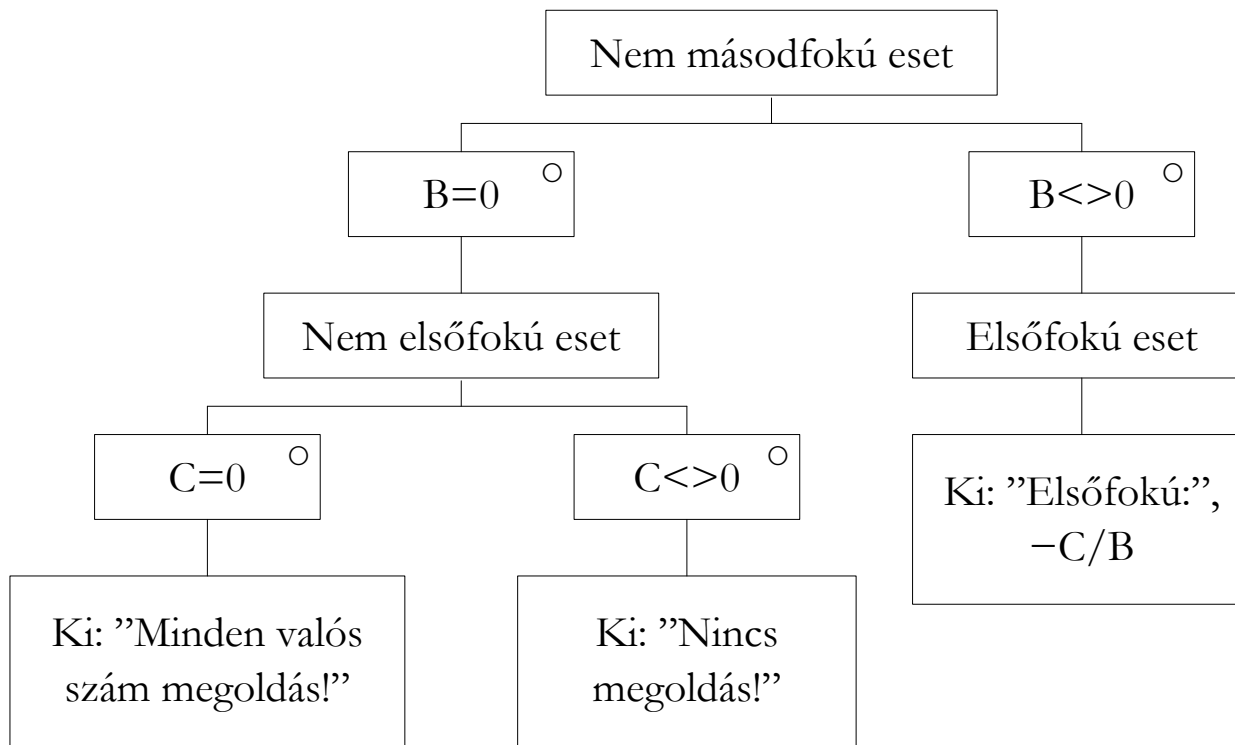
- **Feladat:** Oldjuk meg az  $ax^2+bx+c=0$  másodfokú egyenletet, ahol az  $a, b, c$  együtthatók valós számok!

Funkció	Azonosító	Típus	Jelleg
Az együtthatók	A, B, C	Valós	I
Az egyik valós gyök	X1	Valós	O
A másik valós gyök	X2	Valós	O
A komplex gyökök valós része	V	Valós	O
A komplex gyökök képzetes része	K	Valós	O
A diszkrimináns értéke	D	Valós	M

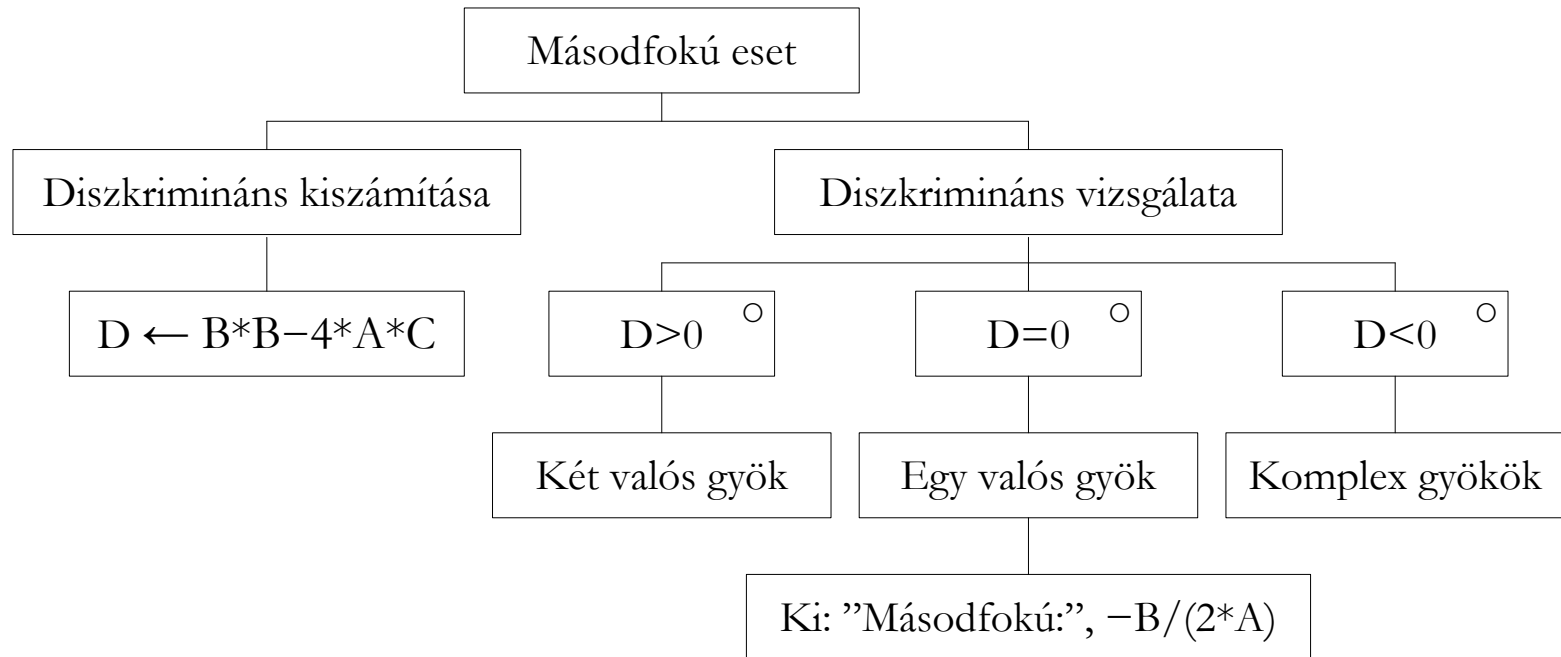
# Strukturált algoritmusok tervezése



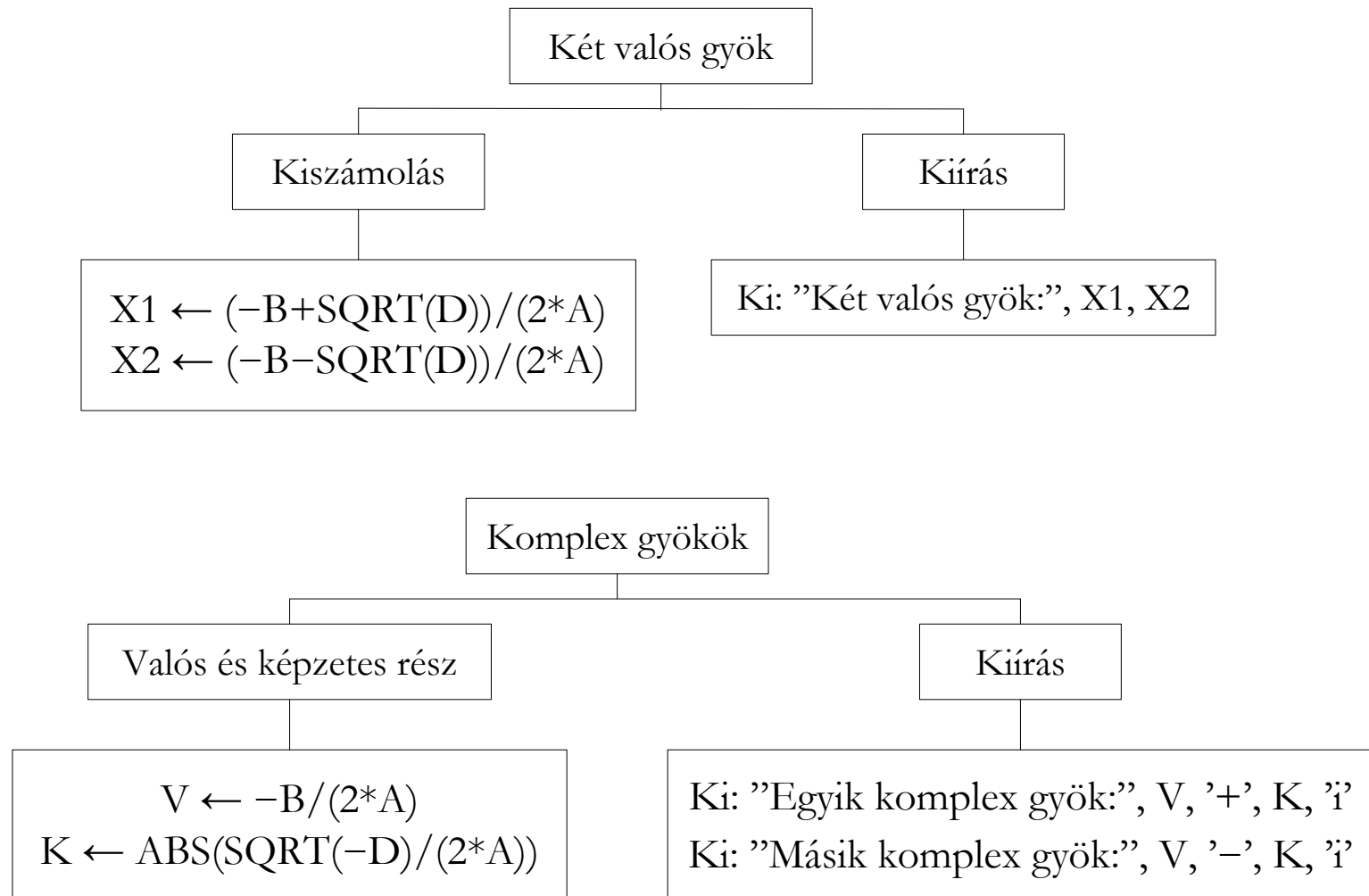
# Strukturált algoritmusok tervezése



# Strukturált algoritmusok tervezése



# Strukturált algoritmusok tervezése



## Strukturált algoritmusok tervezése

```
/* Másodfokú egyenlet megoldása */
```

```
Be: A,B,C
```

```
if A=0
```

```
    /* Nem másodfokú eset */
```

```
    if B=0
```

```
        /* Nem elsőfokú eset */
```

```
        if C=0
```

```
            Ki: "Minden valós szám megoldás!"
```

```
        else
```

```
            Ki: "Nincs megoldás!"
```

```
    else
```

```
        Ki: "Elsőfokú:",  $-C/B$ 
```

```
else
```

```
    /* Másodfokú eset */
```

```
    ...
```





## Strukturált algoritmusok tervezése

...

/\* Másodfokú eset \*/

$D \leftarrow B^2 - 4AC$

**if**  $D > 0$

/\* Két valós gyök \*/

$X_1 \leftarrow \frac{-B + \text{SQRT}(D)}{2A}$

$X_2 \leftarrow \frac{-B - \text{SQRT}(D)}{2A}$

Ki: "Két valós gyök:",  $X_1, X_2$

**else if**  $D = 0$

Ki: "Másodfokú:",  $-B/(2A)$

**else**

/\* Komplex gyökök \*/

$V \leftarrow -B/(2A)$

$K \leftarrow \text{ABS}(\text{SQRT}(-D)/(2A))$

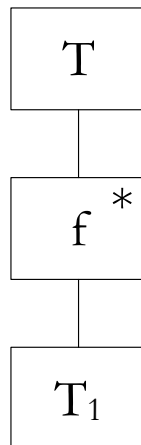
Ki: "Egyik komplex gyök:",  $V, '+', K, 'i'$

Ki: "Másik komplex gyök:",  $V, '-', K, 'i'$



# Strukturált algoritmusok tervezése

- Iteráció:
  - Olyan vezérlőszerkezet, amelyben egy tevékenység ismételten végrehajtható.
- Elöltesztelő iteráció:
  - A  $T$  tevékenység végrehajtása a  $T_1$  tevékenység ismételt végrehajtását jelenti úgy, hogy amíg igaz a ciklust vezérlő feltétel, addig végrehajtjuk a  $T_1$  ciklusmagot. A feltétel értékét a ciklusmag végrehajtása előtt vizsgáljuk.



Strukturadiagram-jelölés

**while**  $f$   
 $T_1$

Pszudokód-jelölés

## Strukturált algoritmusok tervezése

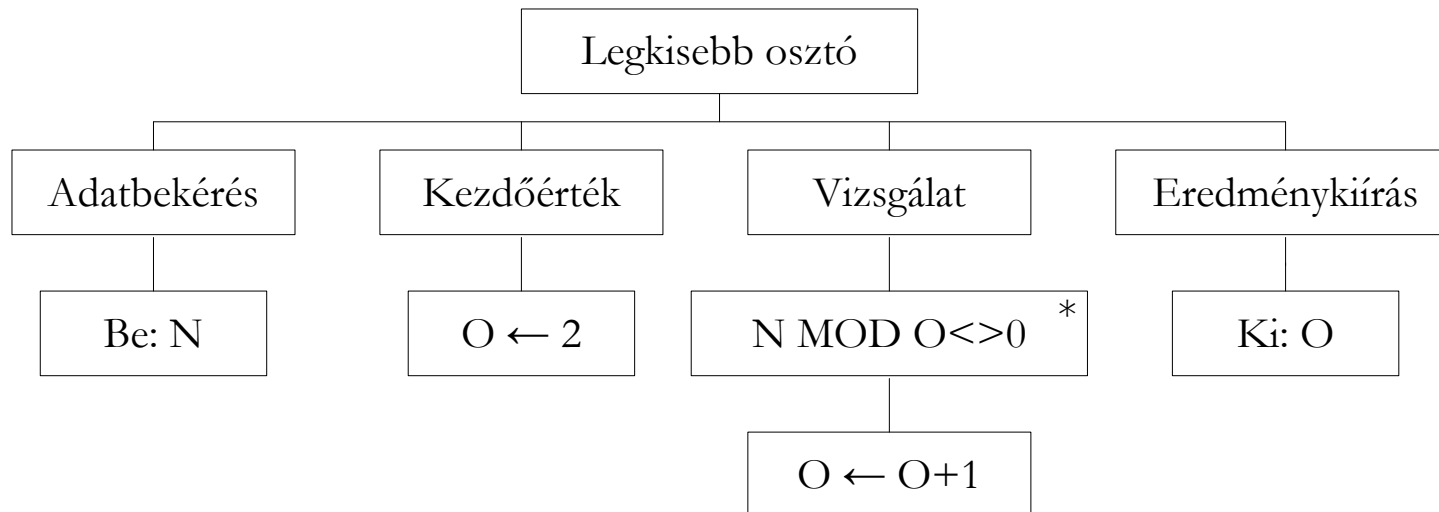
- **Feladat:** Határozzuk meg egy 1-nél nagyobb egész szám 1-től különböző, legkisebb osztóját!

## Strukturált algoritmusok tervezése

- **Feladat:** Határozzuk meg egy 1-nél nagyobb egész szám 1-től különböző, legkisebb osztóját!

Funkció	Azonosító	Típus	Jelleg
A vizsgált szám	N	Egész	I
Az aktuális osztó	O	Egész	M, O

# Strukturált algoritmusok tervezése



/\* Legkisebb osztó \*/

Be: N

O ← 2

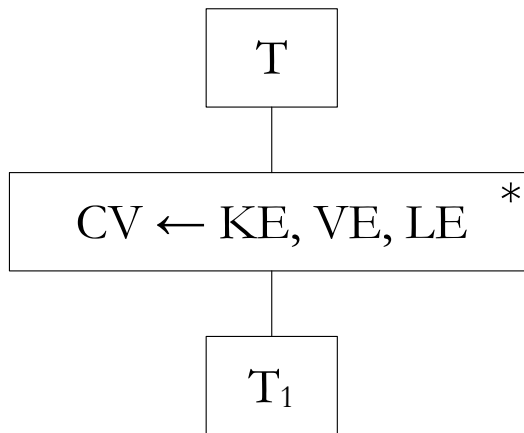
**while** N MOD O <> 0

    O ← O+1

Ki: O

# Strukturált algoritmusok tervezése

- Növekményes iteráció:
  - A T tevékenység végrehajtása a  $T_1$  tevékenység ismételt végrehajtását jelenti úgy, hogy a CV ciklusváltozó a KE kezdőértéktől a VE végértékig lépdel, az LE lépésközzel.
  - A lépésköz alapértelmezése 1, a nulla lépésköz nem megengedett.



Strukturadiagram-jelölés

**for** CV ← KE, VE, LE  
 $T_1$

Pszudokód-jelölés

## Strukturált algoritmusok tervezése

- **Feladat:** Határozzuk meg egy pozitív egész szám faktoriálisát!



## Strukturált algoritmusok tervezése

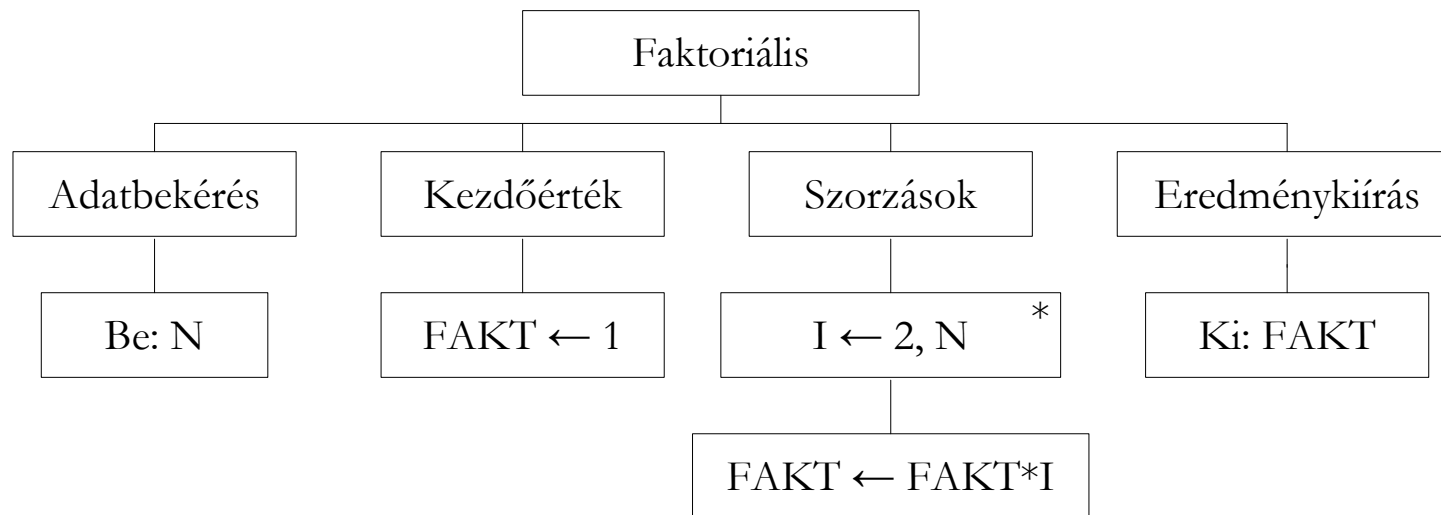
- **Feladat:** Határozzuk meg egy pozitív egész szám faktoriálisát!

Funkció	Azonosító	Típus	Jelleg
N értéke	N	Egész	I
Az aktuális szorzat	FAKT	Egész	M, O
Az aktuális szorzó	I	Egész	M

- **Megjegyzés:**
  - A megoldás programmá írásakor célszerű az eredményt tároló FAKT változót a „legnagyobb” egész típusúra vagy valós típusúra deklarálni.



# Strukturált algoritmusok tervezése



/\* Faktoriális \*/

Be: N

FAKT ← 1

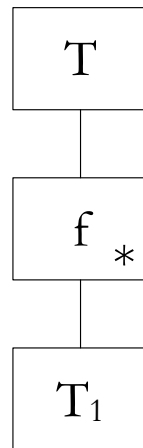
**for** I ← 2, N

    FAKT ← FAKT\*I

Ki: FAKT

# Strukturált algoritmusok tervezése

- Hátultesztelő iteráció:
  - A T tevékenység végrehajtása a  $T_1$  tevékenység ismételt végrehajtását jelenti úgy, hogy amíg a ciklust vezérlő feltétel igazzá nem válik, addig végrehajtjuk a  $T_1$  ciklusmagot. A feltétel értékét a ciklusmag végrehajtása után vizsgáljuk.



Struktúradiagram-jelölés

**repeat**  
 $T_1$   
**until f**

Pszudokód-jelölés

## Strukturált algoritmusok tervezése

- **Feladat:** Kérjünk be karaktereket az Esc billentyű (végjel) leütéséig és írjuk ki minden egyes karakterre a karakter ASCII kódját és azt, hogy angol betű, számjegy vagy egyéb karakter volt-e a megadott karakter!

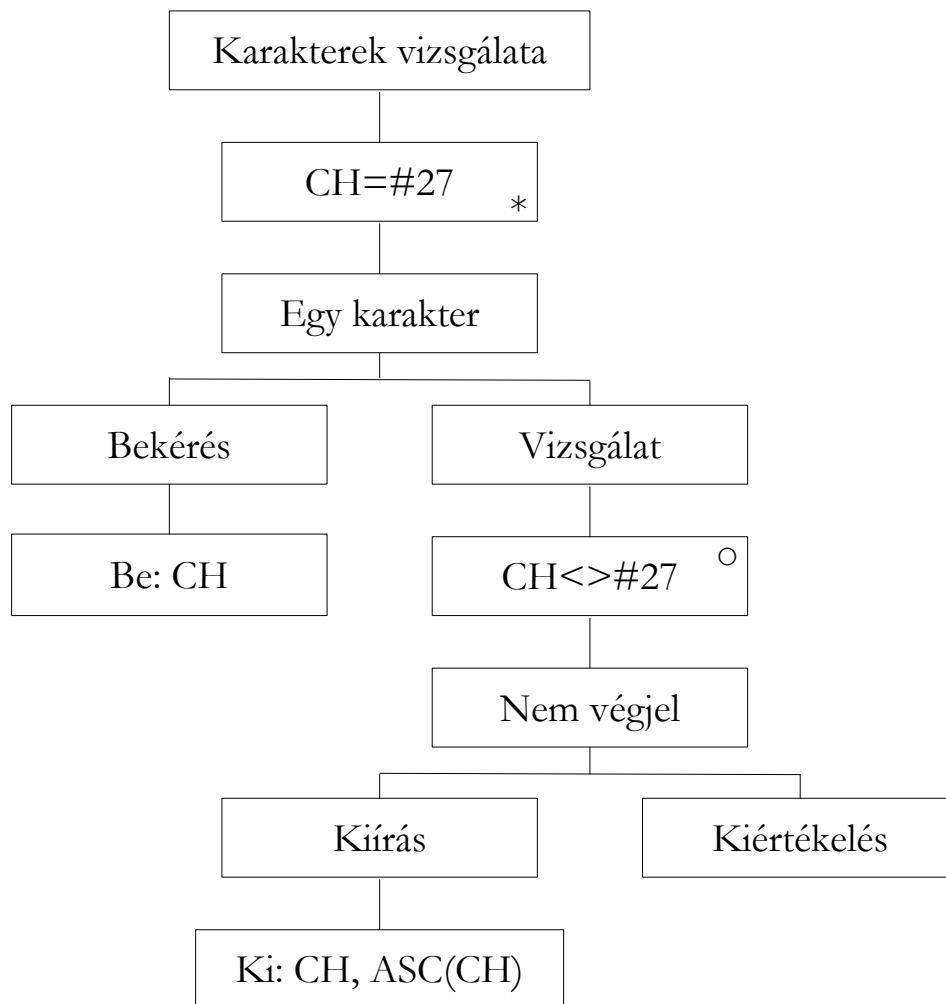
## Strukturált algoritmusok tervezése

- **Feladat:** Kérjünk be karaktereket az Esc billentyű (végjel) leütéséig és írjuk ki minden egyes karakterre a karakter ASCII kódját és azt, hogy angol betű, számjegy vagy egyéb karakter volt-e a megadott karakter!

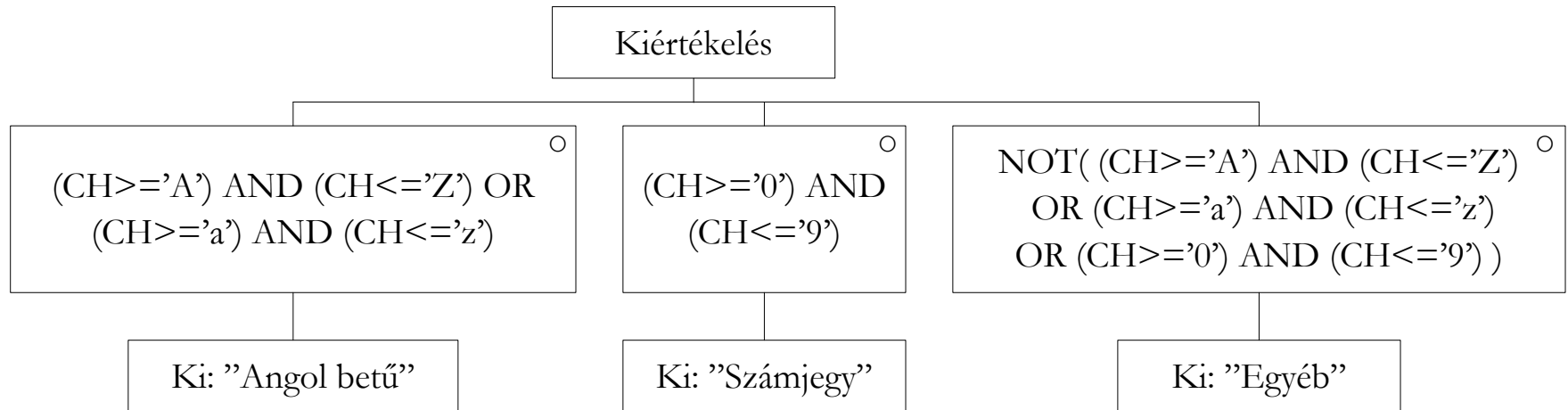
Funkció	Azonosító	Típus	Jelleg
Az aktuális karakter	CH	Karakter	M

- **Megjegyzés:**
  - A változók jellege függ attól, hogy a megoldásra szubrutint készítünk vagy sem.
  - Szubrutinnal történő megoldás esetén a változók jellege a szubrutinra vonatkozóan értendő (I: bemenő paraméter, O: eredmény paraméter, M: munkaváltozó, vagy olyan paraméter, amelynek értéke megváltozik az algoritmus során).
  - A fenti adatszerkezeti táblázat a szubrutinos megoldáshoz igazodik.

# Strukturált algoritmusok tervezése



# Strukturált algoritmusok tervezése



## ■ Megjegyzés:

- A feltételvizsgálatok halmazok és az IN művelet segítségével rövidebben is megadhatók.
- A beolvasást célszerű olyan utasítással programozni, amelyik nem jeleníti meg a leütött karaktert a képernyőn.

## Strukturált algoritmusok tervezése

```
/* Karakterek vizsgálata */  
Ki: "Karakterek vizsgálata (Kilépés:Esc)"  
repeat  
    /* Bekérés */  
    Be: CH  
    if CH<>#27  
        /* Kiírás */  
        Ki: CH,ASC(CH)  
        /* Kiértékelés */  
        if (CH>='A') AND (CH<='Z') OR (CH>='a') AND (CH<='z')  
            Ki: "Angol betű"  
        else if (CH>='0') AND (CH<='9')  
            Ki: "Számjegy"  
        else  
            Ki: "Egyéb"  
until CH=#27
```

