

Algoritmusok

Dr. Iványi Péter

Egyik legrégebbi algoritmus

- i.e. IV század, Alexandria, Euklidész
 - két természetes szám legnagyobb közös osztójának meghatározása
- Tegyük fel, hogy a és b pozitív egész számok és jelöljük (a, b) -vel a és b legnagyobb közös osztóját

Legnagyobb közös osztó

- Ha $a = b * q + r$, akkor $(a , b) = (b , r)$,
így a problémát visszavezethetjük két kisebb szám
legnagyobb közös osztójának meghatározására.
Folytatva az eljárást, az utolsó, 0-tól különböző
maradék a legnagyobb közös osztó.

Legnagyobb közös osztó

$$(360, 225) = ?$$

Az euklidészi algoritmus szerint:

a	b	r
360	= 225 * 1	+ 135
225	= 135 * 1	+ 90
135	= 90 * 1	+ 45
90	= 45 * 2	+ 0

Tehát $(360, 225) = 45$.

A programozás

- Két alapvető koncepció:
 - Mennyiségek, információk közötti kapcsolat leírása

$$x+y$$

- E kapcsolatok kiértékelése, ahol értékeket helyettesítünk nevek helyébe

$$x = 2$$

$$y = 3$$

$$2+3 = 5$$

Műveletek

- Minden adaton lehet egyszerű műveleteket végrehajtani
- Például a számokra definiált:
 - Összeadás, kivonás, osztás, szorzás, ...
- A programozó ezekből az egyszerű műveletekből állítja össze a programot

Információ

- Az információt **adat**ként írjuk le
- Többféle adat van:
 - Egyszerű, (atomi, oszthatatlan), mint a számok
 - Összetett, mint számsorozatok
- Bár az adat reprezentálja az információt, de az értelmezés ránk van bízva.
- Például 13.51 reprezentálhat
 - Hőmérsékletet
 - Időt
 - Távolságot

Programozás, gyakorlati megközelítés

- A probléma felbontása egyszerű a számítógép által is megértett lépésekre
- **A program:** Egy feladat megoldására szolgáló, a számítógép számára értelmezhető utasítássorozat.

Program elemei

- Bemenet
- Kimenet
- Operátorok, műveletek
- (Változók)



Bemenet és kimenet

- Bemenet
 - Billentyűzet
 - Egér
 - Fájl
 - Soros port, stb
- Kimenet
 - Képernyő
 - Fájl
 - Soros port, stb

Változók

- Adatokat változókbán tároljuk
- Név
 - Egy hely ahol tárolunk
- Érték
 - A tárolt adat
- Értékkadás
 - Adott helyre beteszünk egy adatot

Adatok, egyszerű

- Milyen típusú adatot tárolhatunk?
- Számok (bináris szám)
 - Egész
 - Valós
 - Komplex, etc
- Karakterek
 - Bináris számot alakítjuk betűvé, stb.
- Ugyanaz az adat többféleképpen is tárolható
 - 1, '1', "1"

Származtatott, összetett adatok

- Karakterlánc, string, szöveg
 - Vektorok
 - Tömbök, mátrixok
 - Listák
 - Stb.
-
- Programnyelvtől függ!

Műveletek, operátorok

- Adatok manipulálása
- Aritmetikai:
 - $+$, $-$, $*$, $/$
- Relációs, összehasonlító:
 - $>$, $<$, $=$, $<=$, $>=$
- Logikai
 - NEM, ÉS, VAGY

Műveletek sorrendje

- $a=6+12*5$
- Kiértékelési sorrend,
- Hogyan hajtsuk végre a műveleteket?
 - Általában balról jobbra
 - Precedencia
 - Művelet elsőbbsége, erőssége
 - Zárójelezés
 - $a=(6+12)*5$
 - $a=6+(12*5)$

Program, összefoglalva

- Program, több algoritmusból is állhat
- Algoritmus jellemzői:
 - Elvégezhető (elemi, végrehajtható lépésekből áll)
 - Meghatározott (minden lépés pontosan definiált)
 - Véges (véges számú lépés után véget ér)
 - Meghatározott input halmazra érvényes
 - Megfelelő outputot eredményez
 - Egy feladat megoldására szolgál

Algoritmusok, programok tervezése

Programozási módszertan

- 1960-as évek végéig monolitikus programozás
- Jellemzői:
 - Egy programozó egy program
 - A programoknak nincs szerkezete

Programozási módszertan

- A jó program legfontosabb kritériumai
 - jól áttekinthető **szerkezete** van
 - jól dokumentált
 - „**bizonyítható**” módon **azt csinálja, amit kell**

Moduláris programozás

- Oszd meg és uralkodj elv
- **Top-Down dekompozíció**
 - A feladatot részfeladatokra bontjuk, majd azokat további részfeladatokra, míg kezelhető méretű részproblémákhoz nem jutunk
- **Bottom-Up kompozíció**
 - Itt is részfeladatokat oldunk meg, de előre nem tudható, hogy hogyan fognak kapcsolódni egymáshoz

Moduláris programozás

- Előnyök:
 - Részprogramok könnyen áttekinthetők
 - Könnyebben megírható
 - Könnyebben tesztelhető
 - Több modul írható egy időben (párhuzamos problémamegoldás)
 - Könnyebben javítható
 - A modulok szabványosíthatók
 - Modulkönyvtárakban tárolhatók
 - Újrafelhasználhatók

Struktúrált programozás

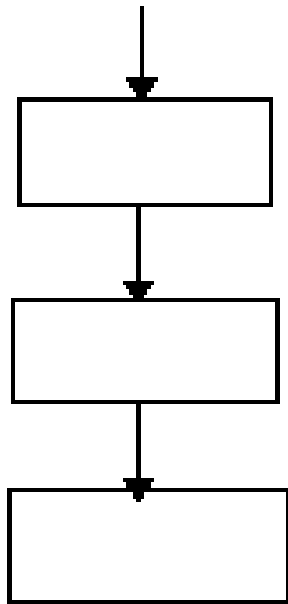
- Dijkstra
- Top-Down dekompozíciót egészíti ki
 - Az **eredeti feladat részfeladataira bontása**, keletkezik egy absztrakt program, mely egy absztrakt számítógépen működik és mivel az eredeti specifikációból indulunk ki bizonyítható módon működik a program.
 - **Finomítás**, mely csökkenti az absztrakciót (egy részprobléma kifejtése) újabb absztrakt gépen újabb utasításkészlet mellett megint bizonyíthatóan működik a program.
 - **További finomítások**, míg egy konkrét gép konkrét utasításkészletéig eljutunk.

Struktúrált programozás

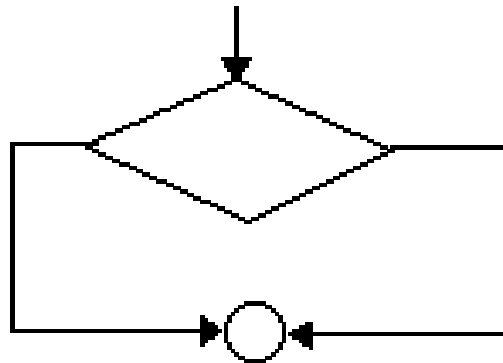
- Végeredmény:
 - egy bizonyítottan helyes program
- Probléma:
 - elfogadhatatlanul nagyobb munka árán.
- Megoldás:
 - A bizonyítás lépéseit elhagyva egy megközelítőleg jó program készítése, melyeket tesztekkel lehet ellenőrizni.

Boehm, Jackopini 1964

- Minden algoritmikus program vezérlési szerkezete leírható 3 vezérlőszerkezet segítségével.

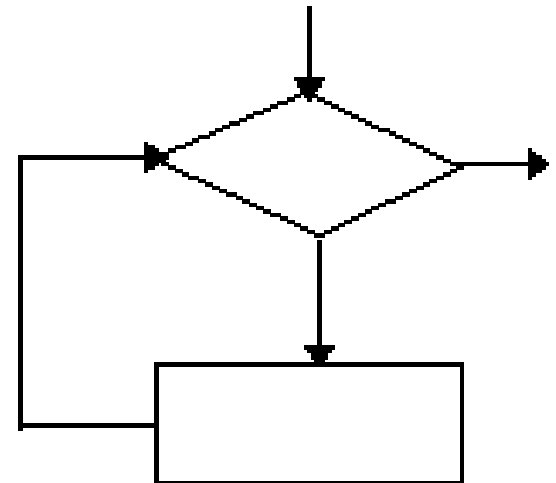


Szekvencia



Szelekció

elágazás, döntés



Iteráció

ciklus, ismétlés

Mills 1968

- Bebizonyítja, a Boehm és Jackopini elméletét az alábbi megkötésekkel:
 - Minden program szerkezete egy szekvencia
 - Minden szekvencia elemnek egy belépési és egy kilépési pontja lehet. (Egyik rész kimenete a másik rész bemenete.)
 - Minden szekvencia belülről tetszőlegesen struktúrázható.

Algoritmusok ábrázolása, leírása

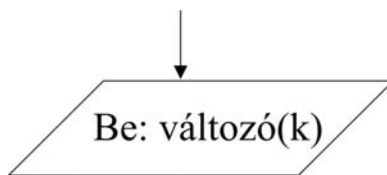
- Pszeudo kód
 - Mondatszerű leírás
- Folyamatábra
 - Blokkdiagram
- Struktogram
 - Egyetlen téglalap tagolása, amely a teljes feladat részekre bontását jelenti.
- ...

Folyamatábra

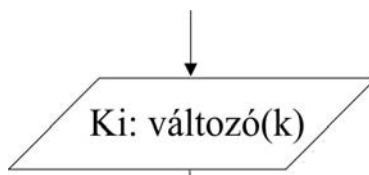
- Kezdés



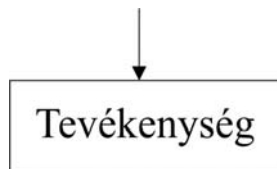
- Bemeneti adat



- Kimeneti adat

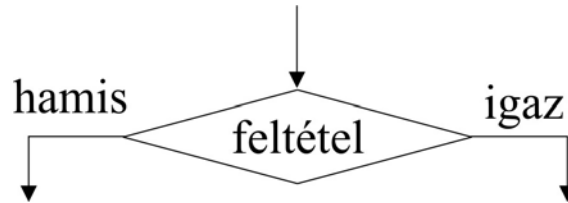


- Tevékenység

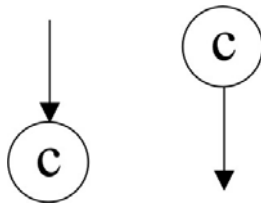


Folyamatábra

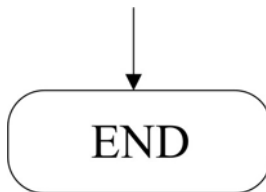
- Elágazás



- Címke

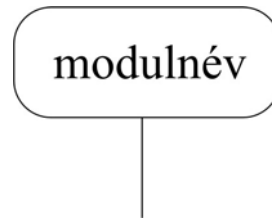


- Vége

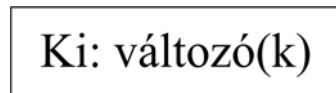


Struktogram

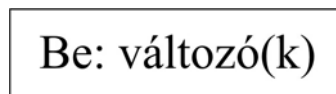
- Program eleje



- Bemeneti adat



- Kimeneti adat

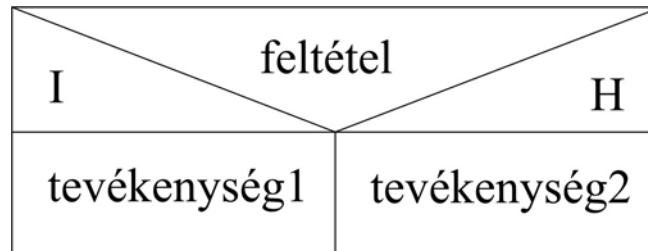


Struktogram

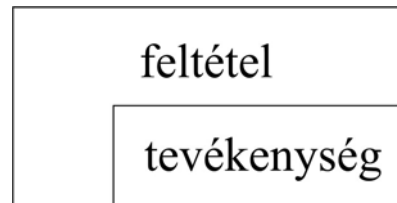
- Tevékenység



- Elágazás



- Ciklus



Program végrehajtás

- Egyik utasítást a másik után hajtja végre a számítógép
- Vezérlő szerkezetek: eltérés ettől a sorrendtől

Vezérlő szerkezetek

- Ugrás
- Feltételes elágazás
- Többszörös elágazás
- Számláló ciklus
- Elöltesztelő ciklus
- Hátultesztelő ciklus
- (Alprogramok)

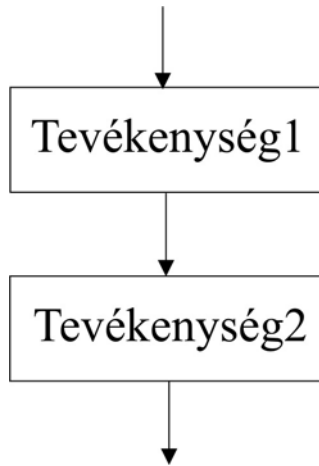
Ugrás

- A program egy megadott utasítánál folytatódik, nem a következőnél
- Nehezen követhető struktúrát eredményez
- Nem használjuk!!!

Alprogram

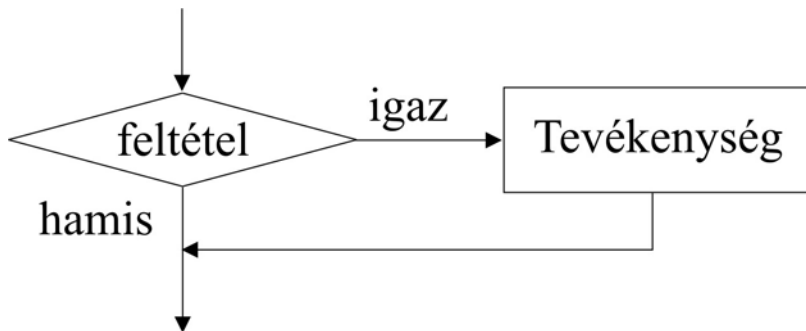
- Ismétlődő feladat
- Alfeladatok elkülönítésére
- Szubrutin, eljárás, függvény

Programszerkezetek, folyamatábra

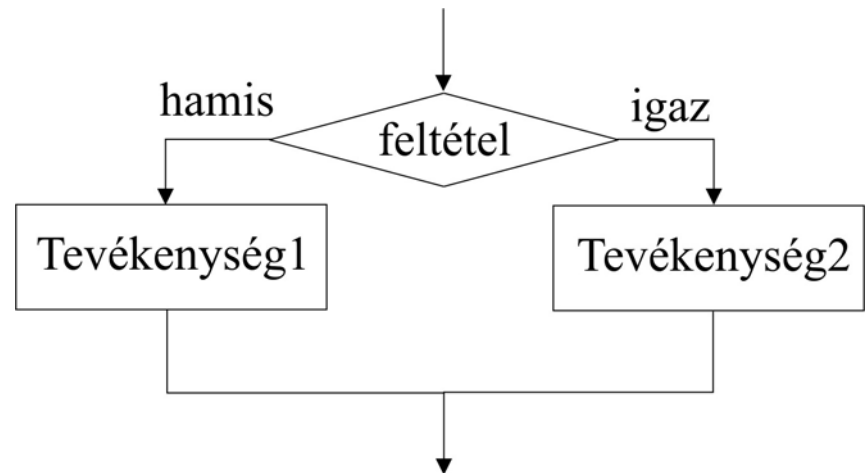


Szekvencia

Egyágú szelekció

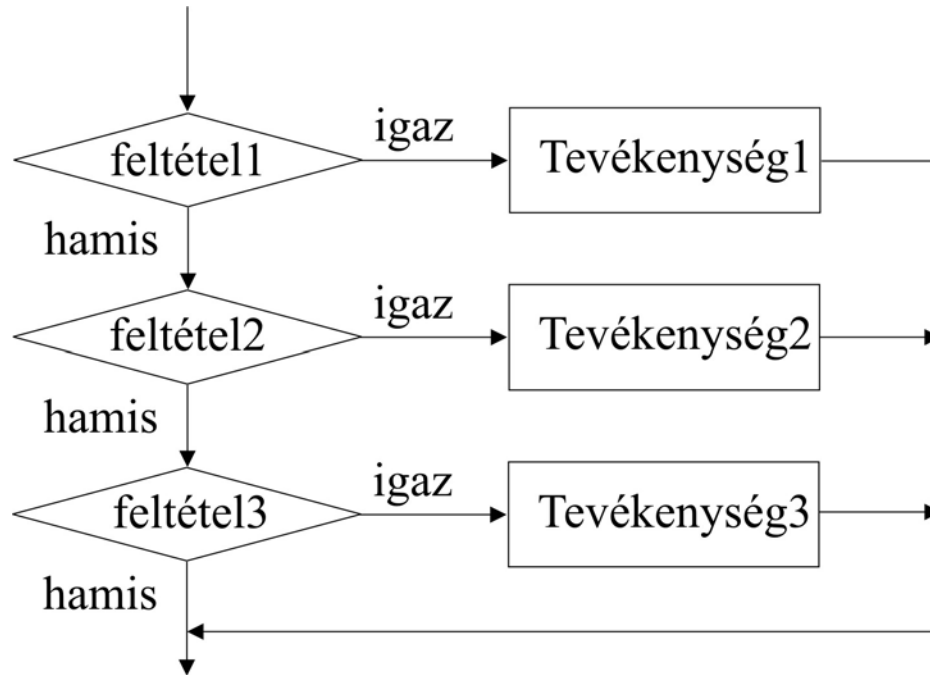


Kétágú szelekció

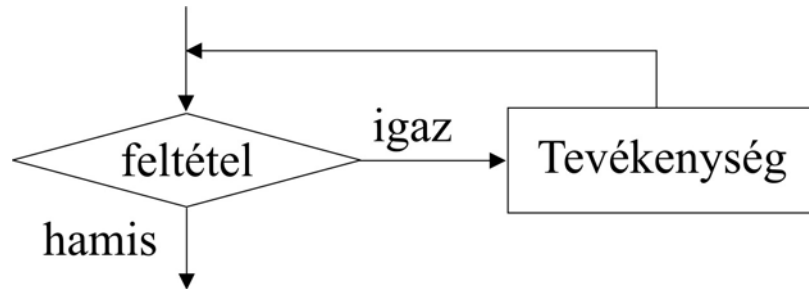


Programszerkezetek , folyamatábra

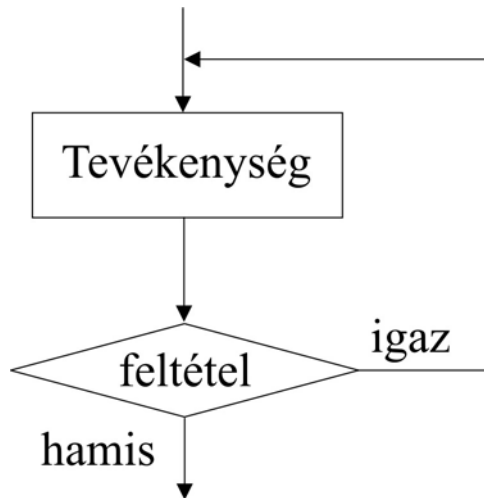
Többágú szelekció



Programszerkezetek , folyamatábra



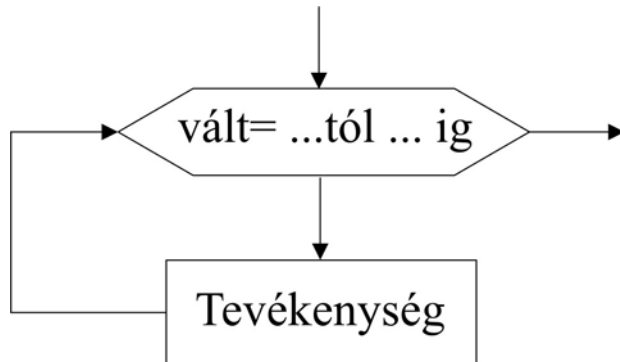
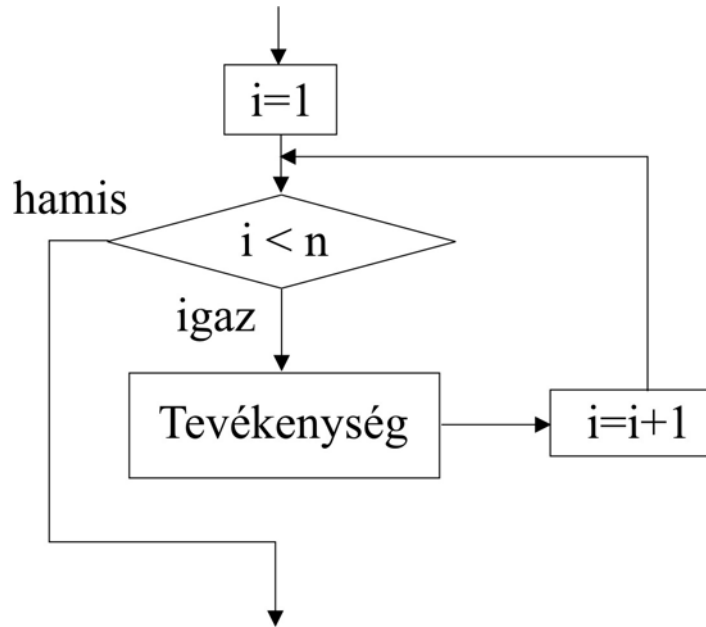
Elöltesztelő ciklus



Hátultesztelő ciklus

Programszerkezetek , folyamatábra

Növekményes ciklus



Programszerkezetek, struktogram

tevékenység1
tevékenység2

szekvencia

I	feltétel	H
tevékenység1	—	

Egyágú szelekció

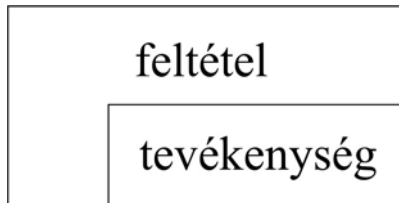
I	feltétel	H
tevékenység1	tevékenység2	

Kétágú szelekció

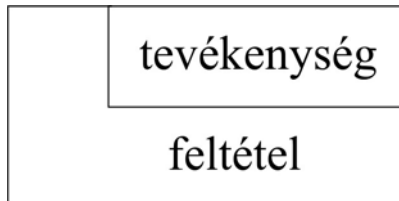
Programszerkezetek, struktogram

feltétel1	feltétel2		feltétel
tevékenység1	tevékenység2	...	tevékenység

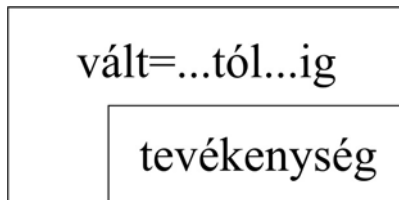
Többágú szelekció



Elöltesztelő ciklus

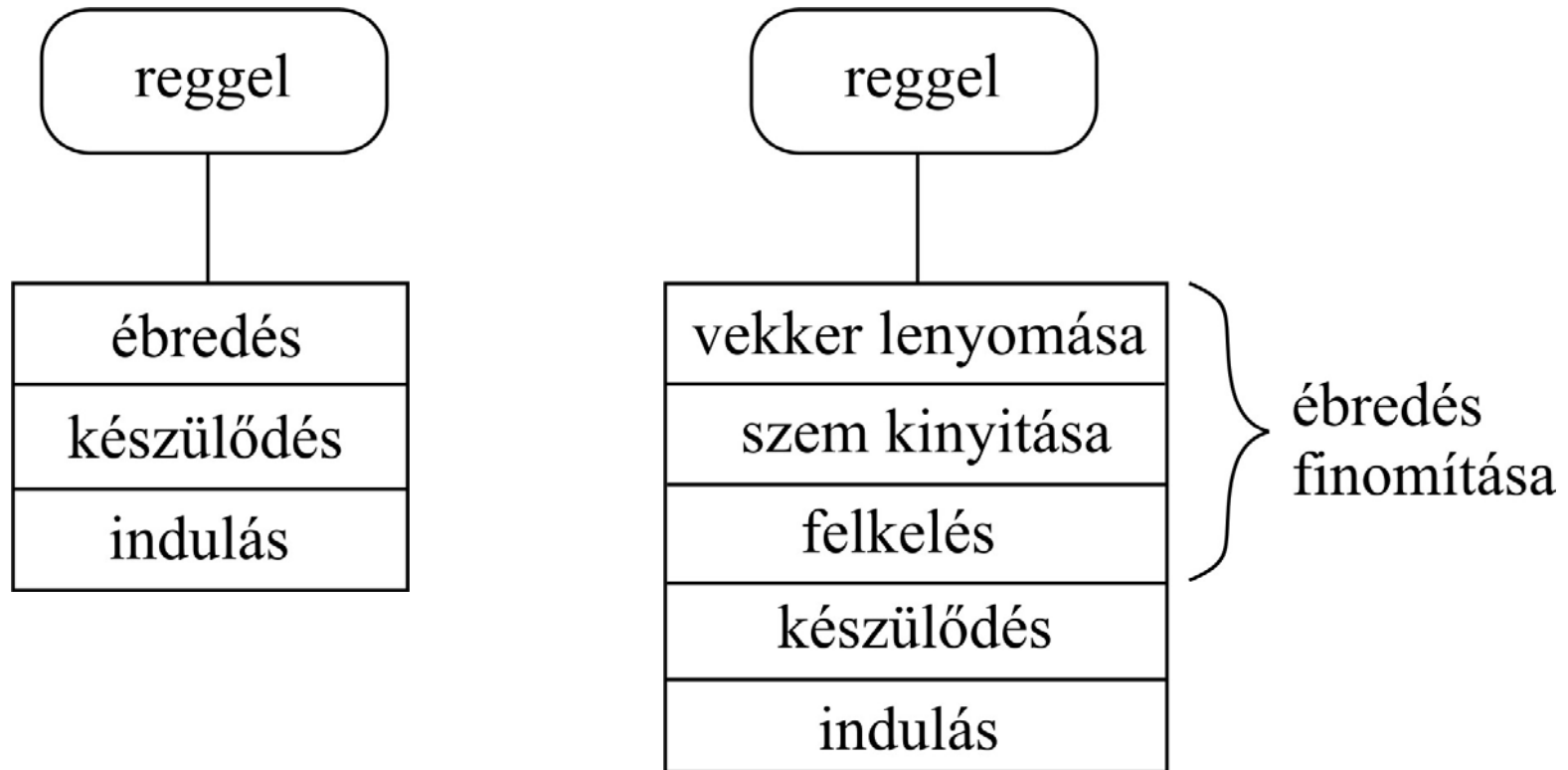


Hátultesztelő ciklus



Növekményes ciklus

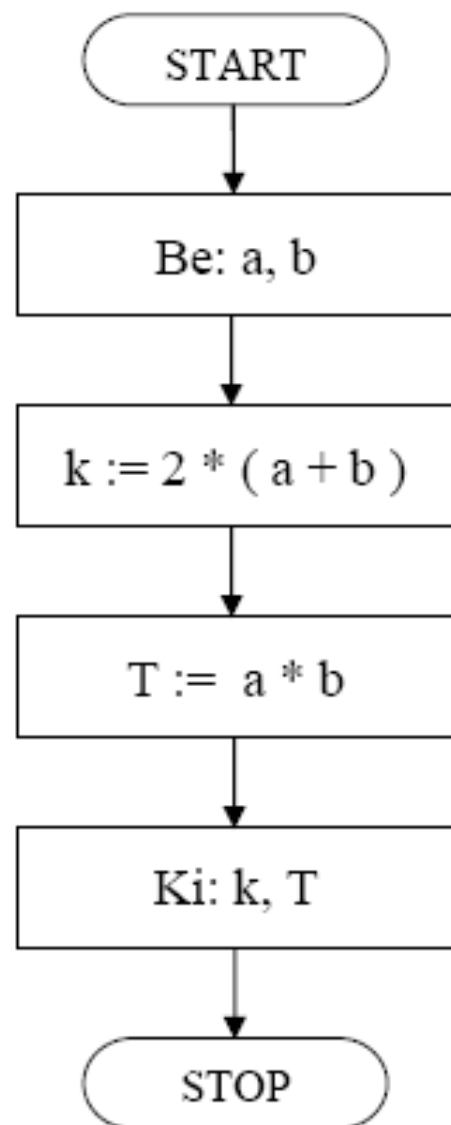
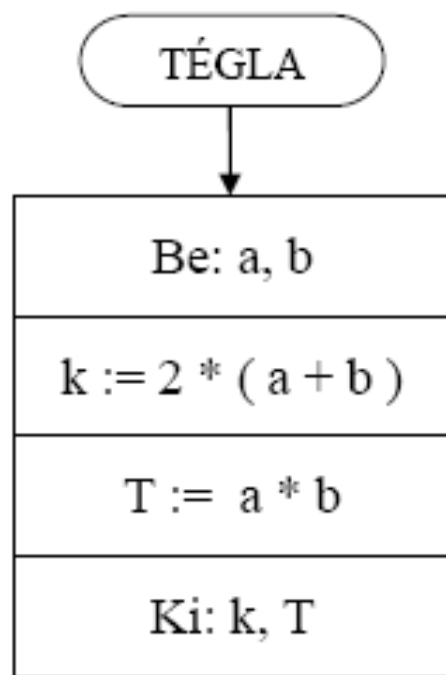
Szekvencia, példa



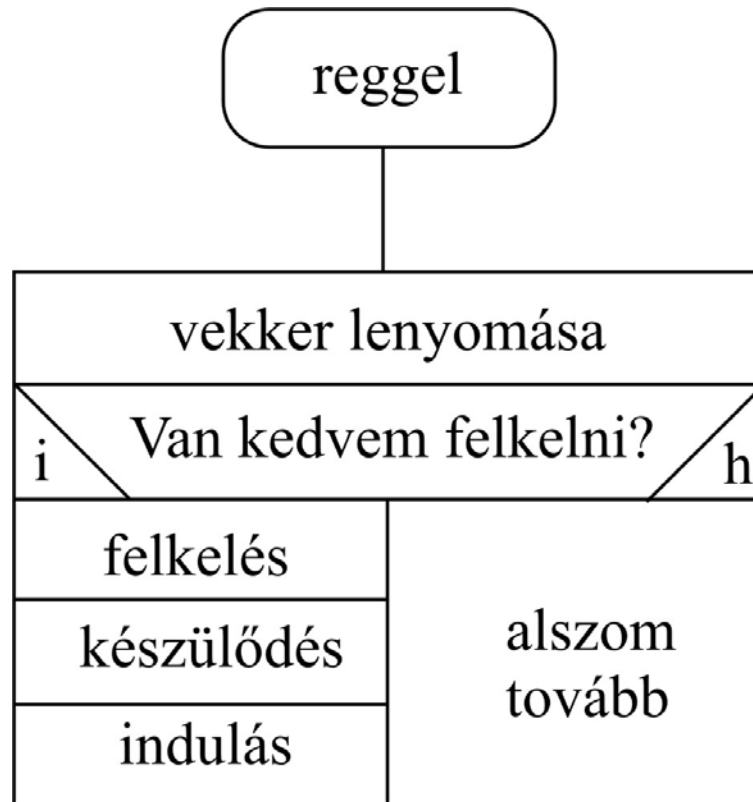
PÉLDA A SZEKVENCIÁRA: Egy téglalap kerületének és területének kiszámítása

Változók:

- a: **Valós** a téglalap egyik oldala
- b: **Valós** a téglalap másik oldala
- k: **Valós** a téglalap kerülete
- T: **Valós** a téglalap területe

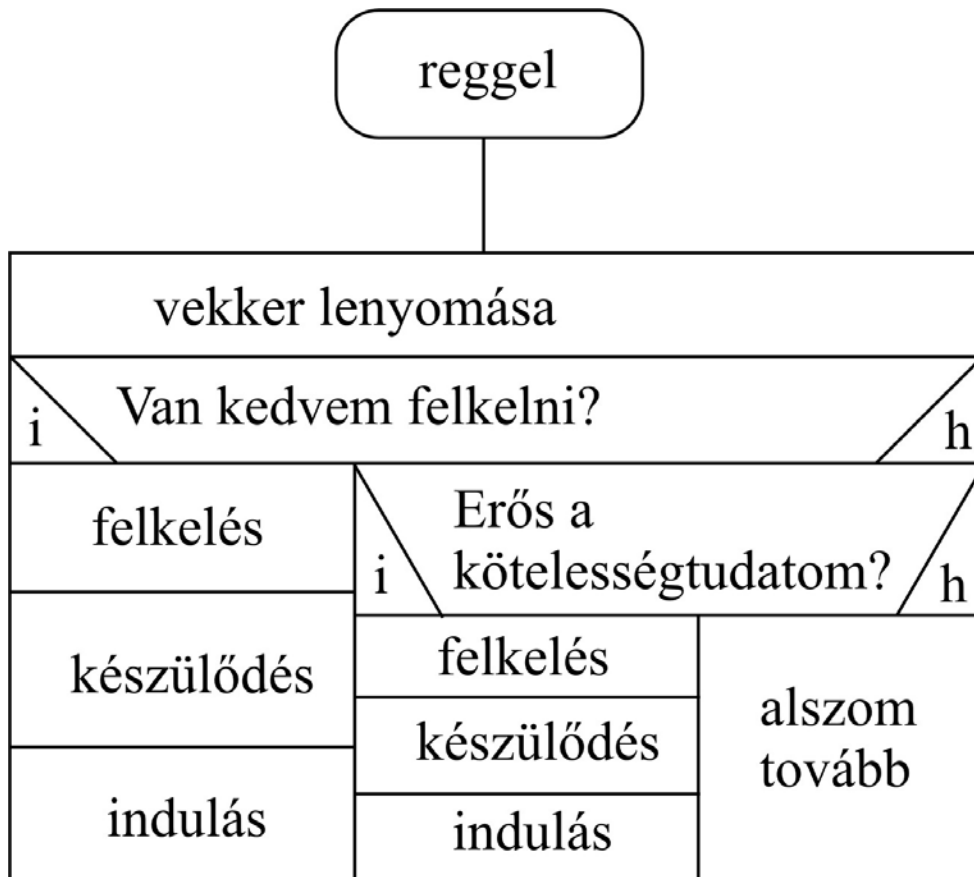


Szelekció, példa 1.



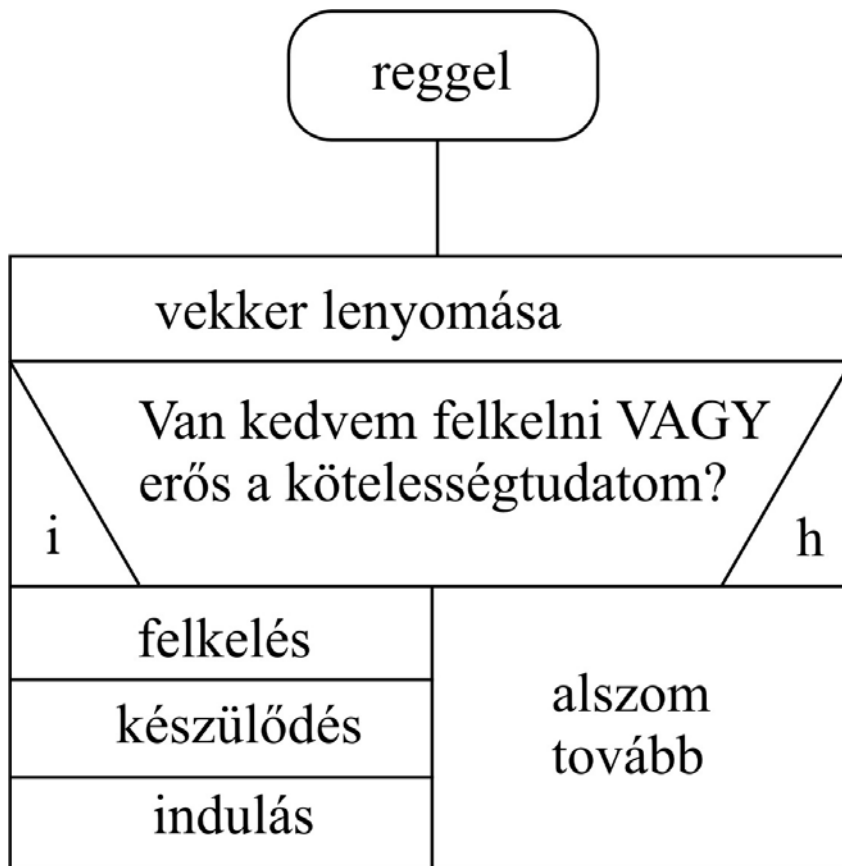
Szelekció, példa 2.

Szelekció bármelyik ágába újabb szelekció is tehető



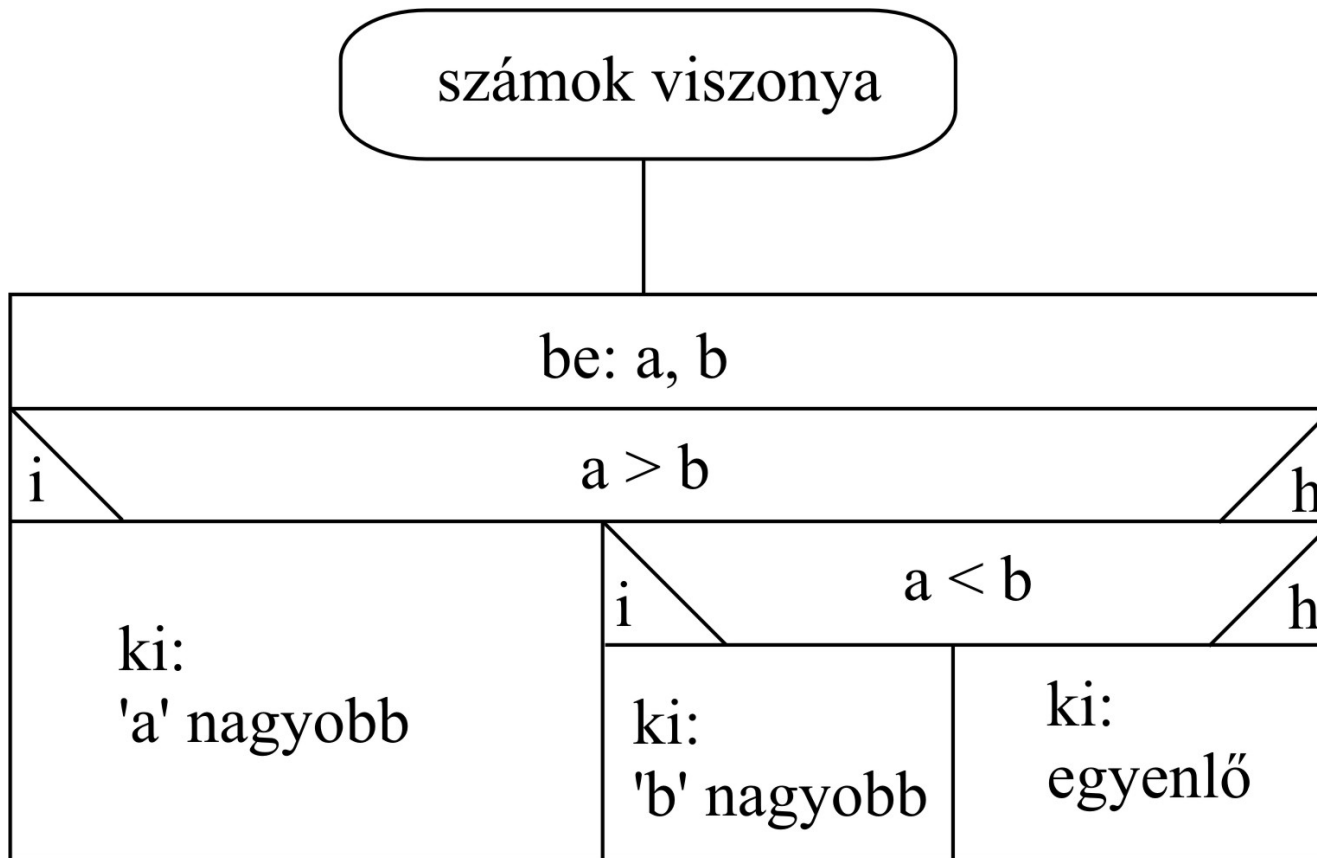
Szelekció, példa 3.

Feltétel össze is vonható



Szelekció, példa 4.

Számok egymás közti viszonya



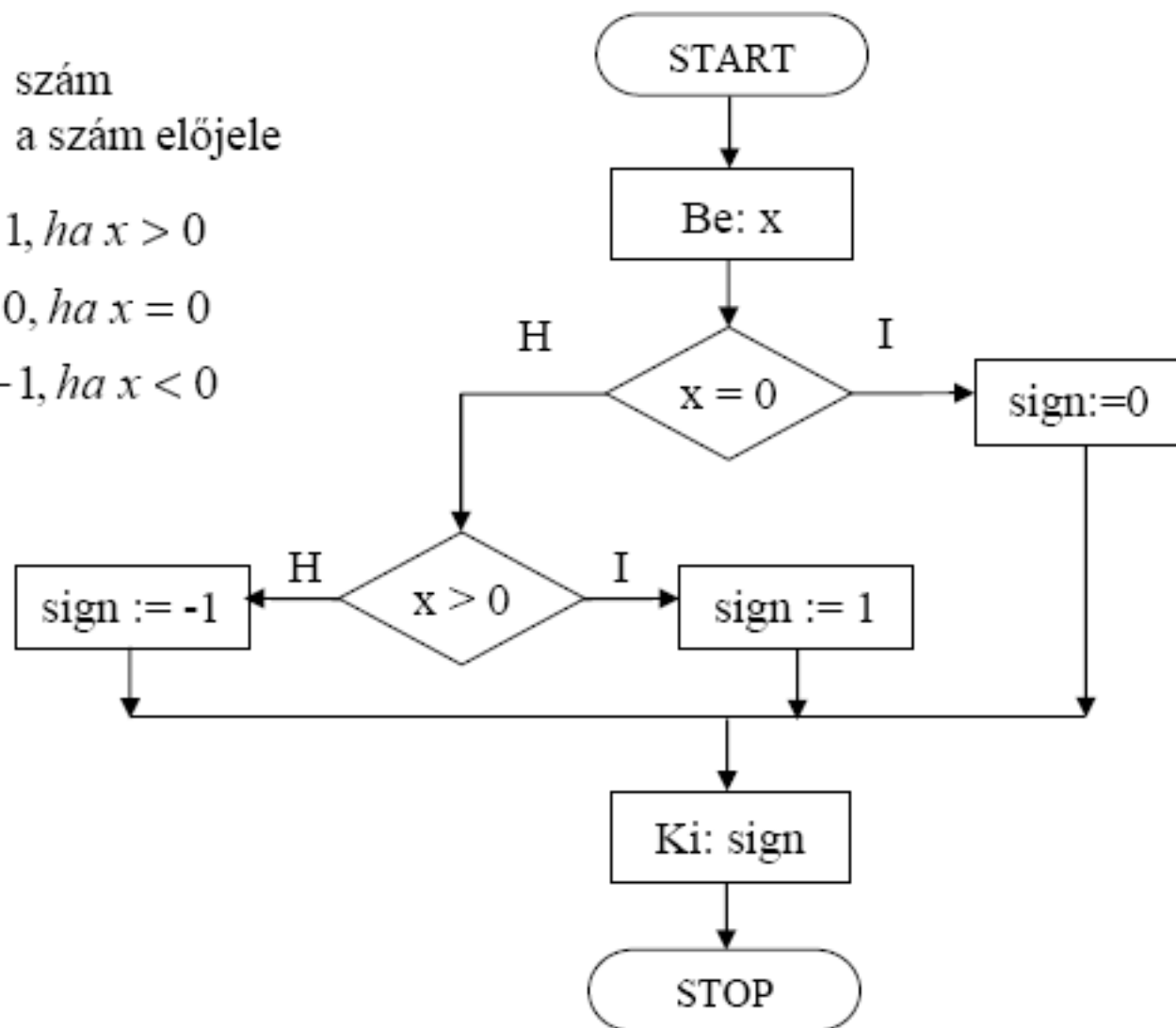
PÉLDA A SZELEKCIÓRA: Egy számelőjele az előjel függvény szerint

Változók:

x: **Valós** szám

sign: Valós a szám előjele

$$\text{sign}(x) := \begin{cases} 1, & \text{ha } x > 0 \\ 0, & \text{ha } x = 0 \\ -1, & \text{ha } x < 0 \end{cases}$$



Többszörös szelekció, példa

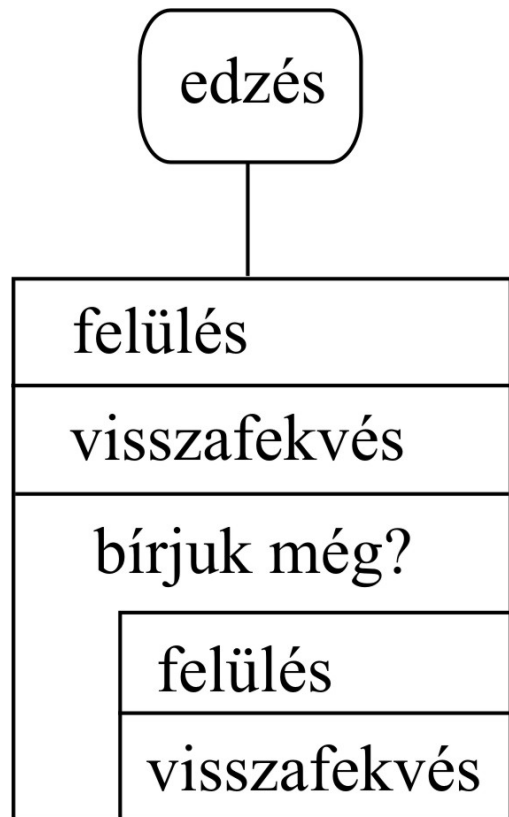
ruhaválasztás

be: t (hőmérséklet)

be: t (hőmérséklet)					
$t \leq -2$	$-2 < t \leq 5$	$5 < t < 15$	$15 < t \leq 20$	$20 < t \leq 30$	$30 < t$
ki: bunda	ki: pulcs és dzseki	ki: dzseki	ki: pulcsi	ki: nyári ruha	ki: fürdőruha

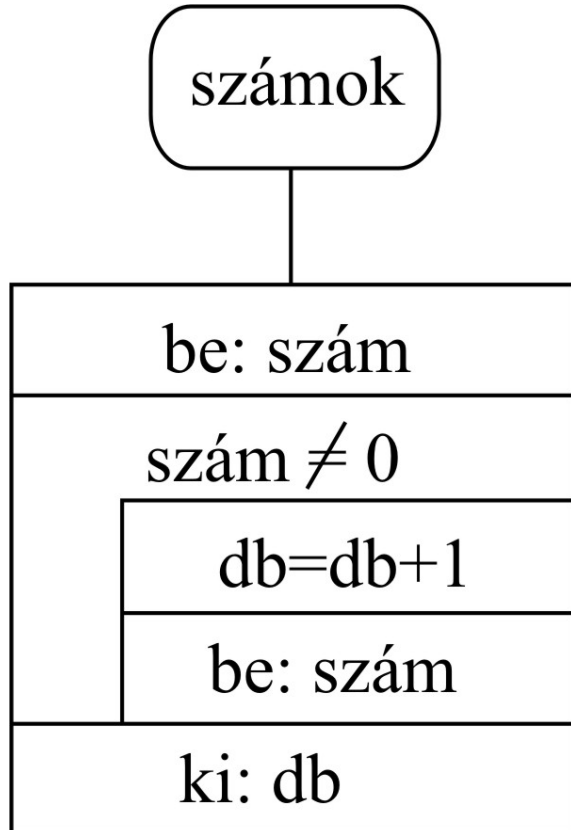
Elöltesztelő ciklus, példa 1.

Addig végzünk felülést amíg bírjuk.



Elöltesztelő ciklus, példa 2.

Egész számokat olvassunk be mindaddig, amíg 0-t nem adnak be. Ha 0-t kapunk, kiírjuk, hány darab számot olvastunk be a 0 kivételével, és vége a feldolgozásnak.



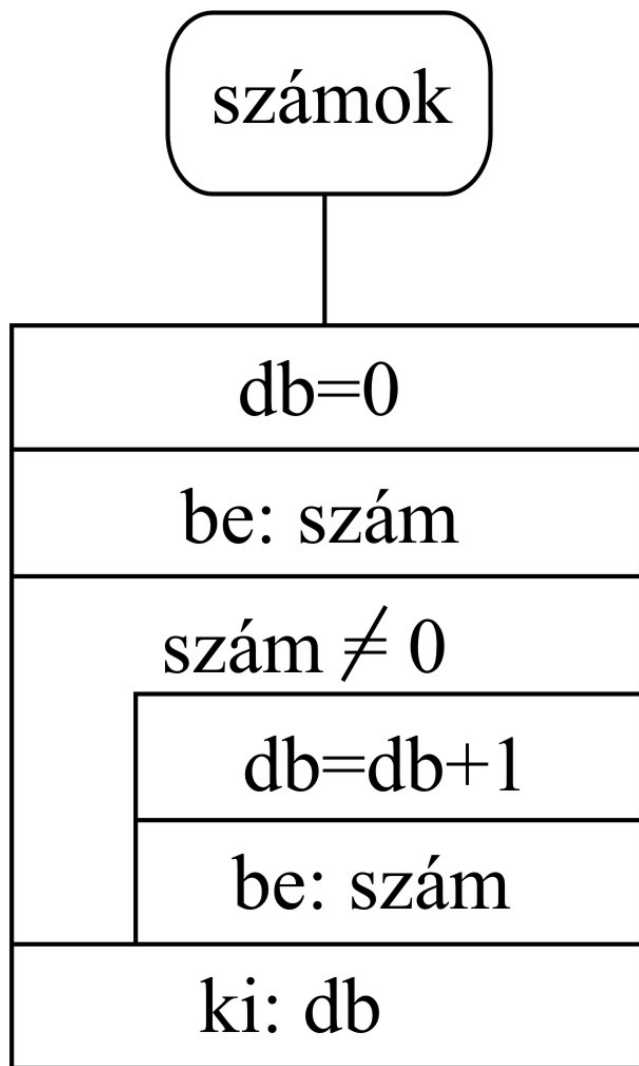
Mit fog kinyomtatni?

Mi a probléma itt?

be	db
	2345
33	2378
5	2383
12	2395

→ 2395

Elöltesztelő ciklus, példa 3.

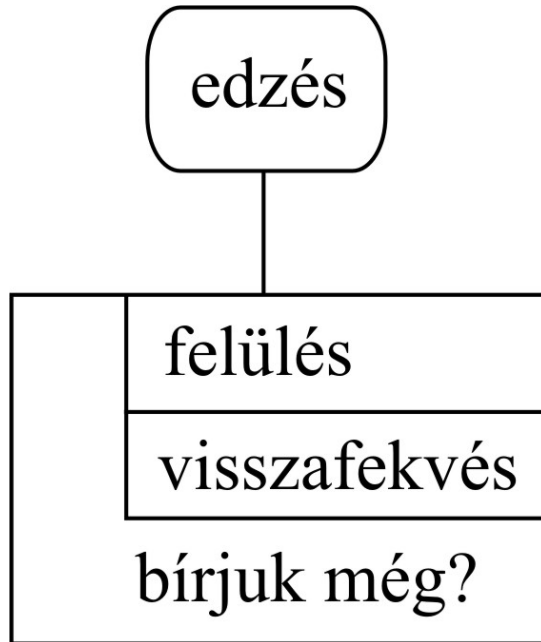


A változókat a legtöbb programozási nyelven inicializálni kell, vagyis kezdeti értéket kell beállítani!

be	db
	0
33	33
5	38
12	50

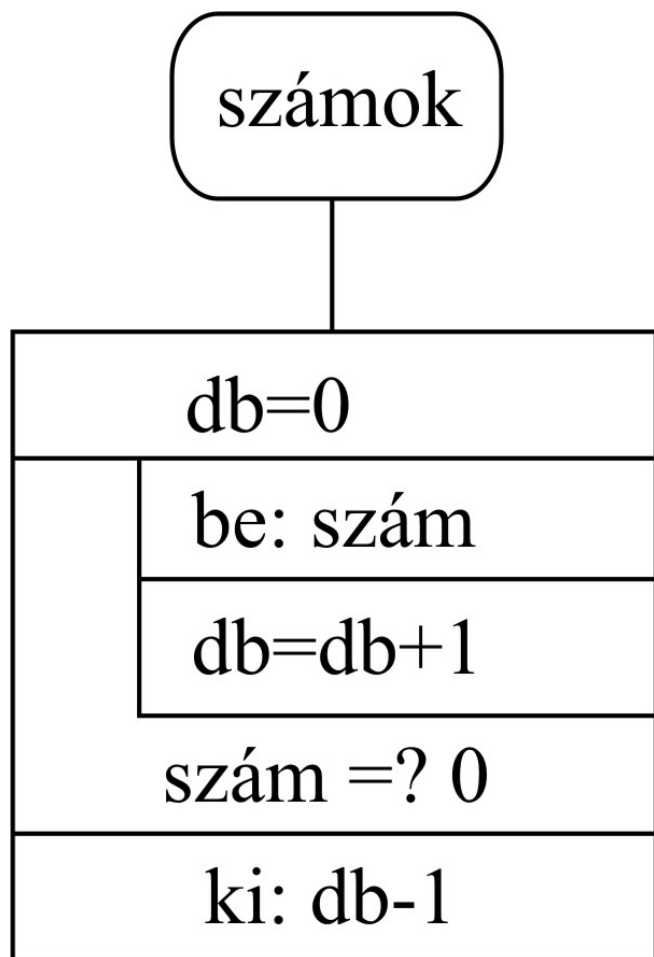
→ 50

Hátultesztelő ciklus, példa 1.



Az előző testgyakorlási példa újra.

Hátultesztelő ciklus, példa 2.

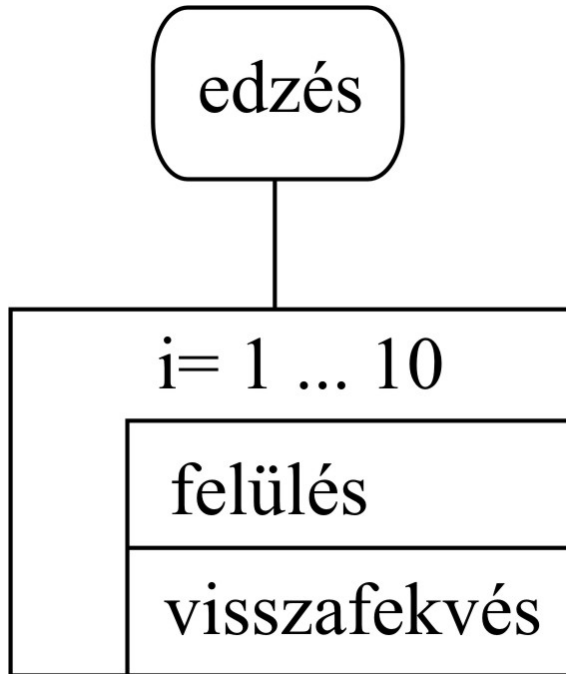


Csak egy helyen olvasunk be számot!

Megváltozott a ciklus feltétel.

Kimeneti adat is megváltozott, mivel az utolsó nullát nem akarjuk hozzászámolni.

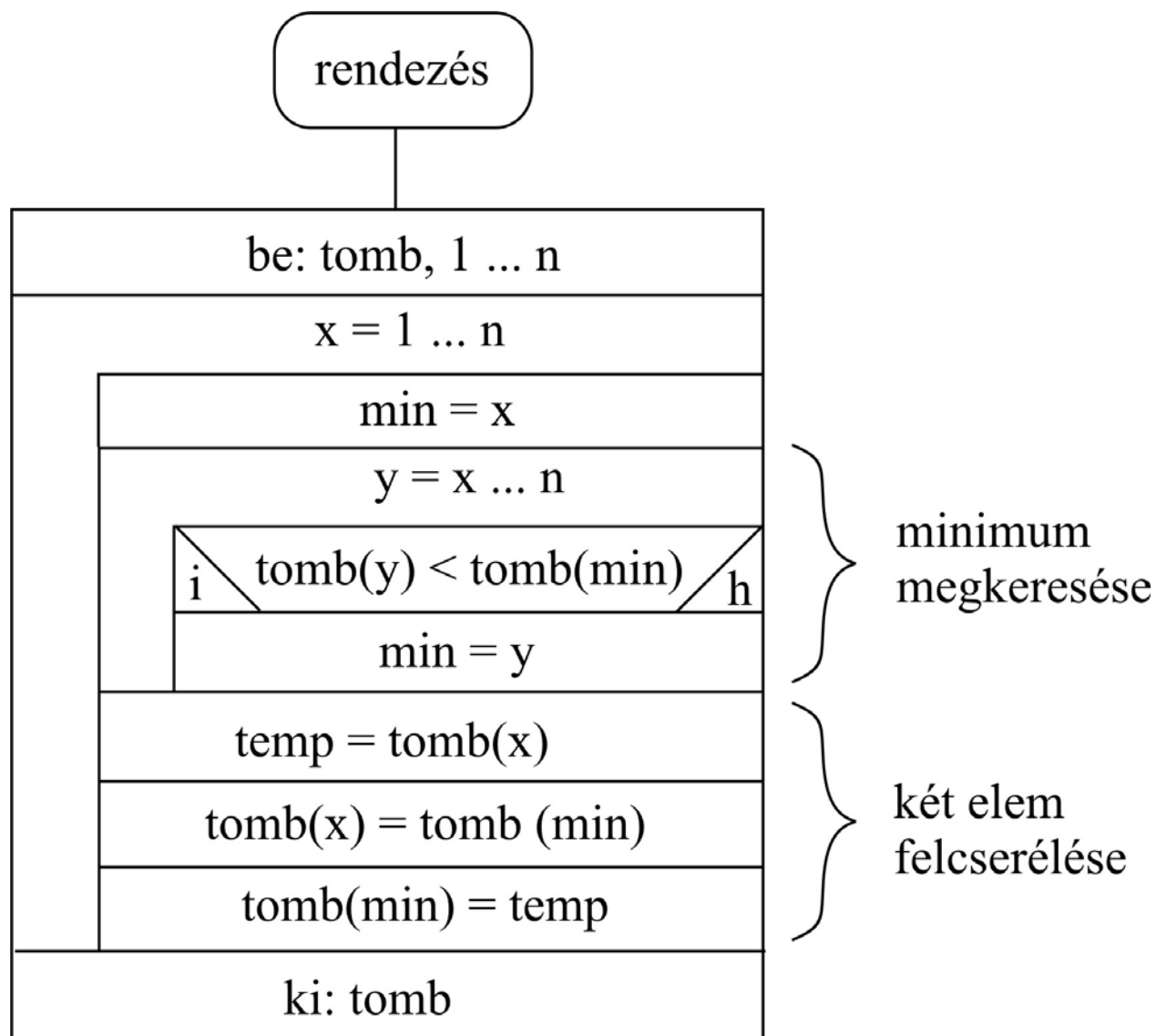
Számláló ciklus, példa



Számláló ciklus

Csak 10 felülést akarunk végezni.

Kiválasztásos rendezés, 1



Rendezés 2

index	1	2	3	4	5	6
tömb	5	2	6	1	3	4

$x = 1$

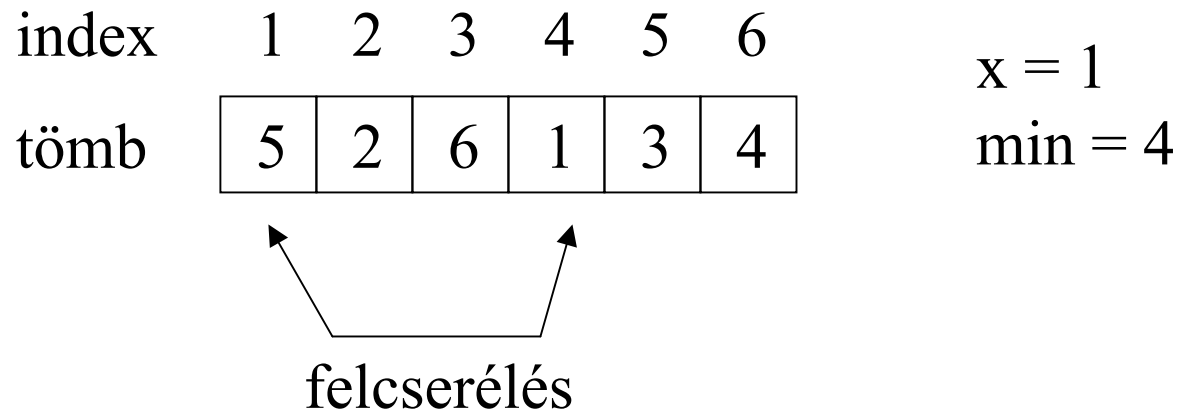
$\text{min} = 1$

$y = 1 \dots 6$

y	min
1	1
2	2
3	2
4	4
5	4
6	4

→ $\text{min} = 4$

Rendezés 3



Az eredmény:

index	1	2	3	4	5	6
tömb	1	2	6	5	3	4

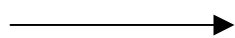
Rendezés 4

index	1	2	3	4	5	6
tömb	1	2	6	5	3	4

$x = 2$
 $\text{min} = 2$

$y = 2 \dots 6$

y	min
2	2
3	2
4	2
5	2
6	2




$\text{min} = 2$

Önmagával kellene
felcserélni, nincs változás

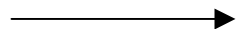
Rendezés 5

index	1	2	3	4	5	6
tömb	1	2	6	5	3	4

$x = 3$
 $\text{min} = 3$


 $y = 3 \dots 6$

y	min
3	3
4	4
5	5
6	5

 $\text{min} = 5$

Rendezés 6

index	1	2	3	4	5	6
tömb	1	2	6	5	3	4

$x = 3$
 $\text{min} = 5$



felcserélés

Az eredmény:

index	1	2	3	4	5	6
tömb	1	2	3	5	6	4

Rendezés 7

index	1	2	3	4	5	6
tömb	1	2	3	5	6	4

$x = 4$
 $\text{min} = 4$

$y = 4 \dots 6$

y	min
4	4
5	4
6	6

→ $\text{min} = 6$

Rendezés 8

index	1	2	3	4	5	6
tömb	1	2	3	5	6	4

$x = 4$
 $\text{min} = 6$




felcserélés

Az eredmény:

index	1	2	3	4	5	6
tömb	1	2	3	4	6	5

Rendezés 9

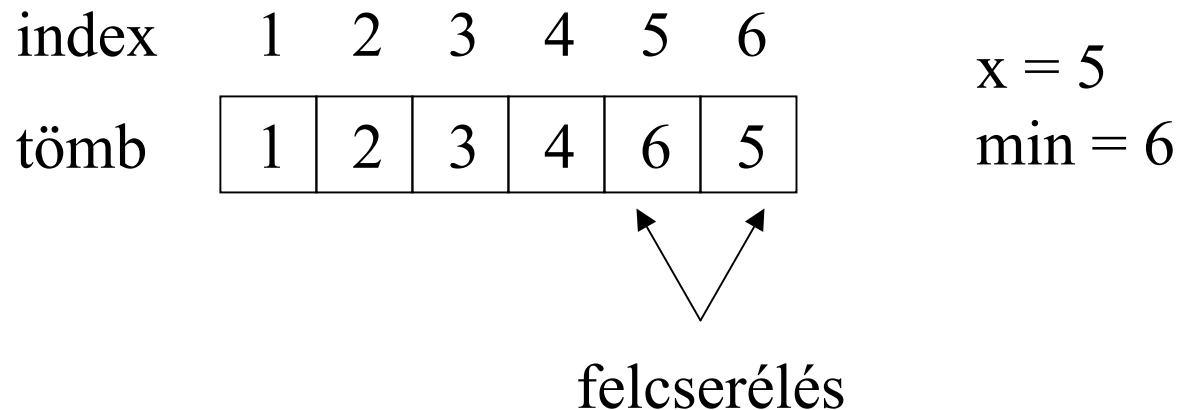
index	1	2	3	4	5	6	
tömb	1	2	3	4	6	4	$x = 5$ $\min = 5$


 $y = 5 \dots 6$

y	min
5	5
6	6

→ $\min = 6$

Rendezés 10



Az eredmény:

index	1	2	3	4	5	6
tömb	1	2	3	4	5	6

Rendezés bonyolultsága

- Alapból feltételezzük hogy az algoritmus $O(1)$. „Szinte semmit nem csinál”, azonnal visszatér.
- Ezután keressük meg a legbonyolultabb részt.
- A külső ciklus n -szer fut le tehát $O(n)$.
- A belső ciklus is lényegében n -szer fut le. Bár ez változó hiszen x -től függ.
- Így az átlagos hatékonyság: $n/2$. De mivel a konstansokat nem vesszük figyelembe, ezért $O(n)$.
- A kettőt összeszorozva: $O(n^2)$ kapunk

Példa 1, probléma

Bálint gazda sertést, kecskét, juhot vásárolt, összesen 100 állatot, pontosan 100 aranyért. A sertés darabja három és fél arany, egy kecske ára egy és egyharmad arany, egy juh ára fél arany. Határozzuk meg, hány darabot vett mindegyik állatból!

Változók és feltételek

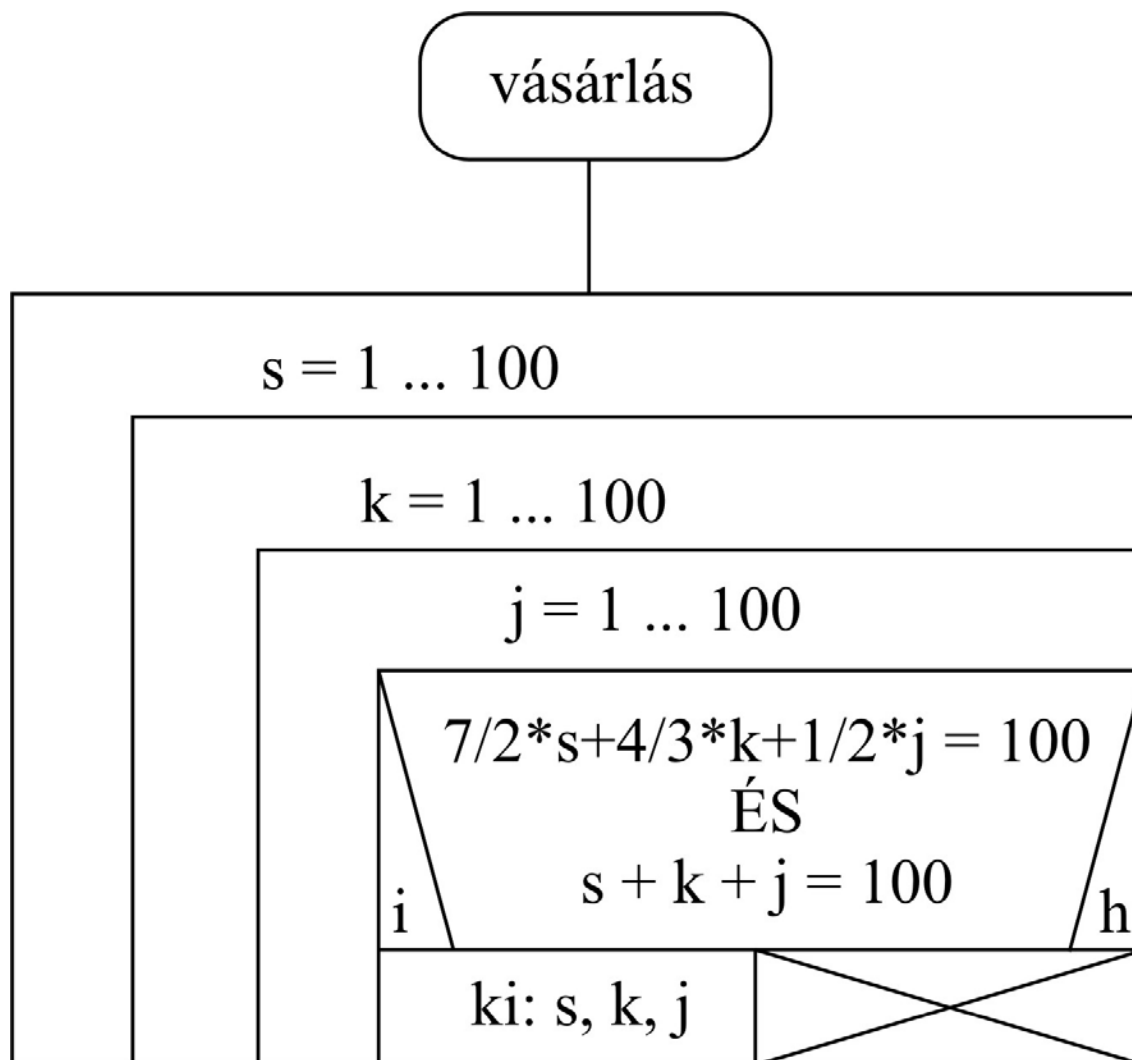
- Változók:

- s : sertések száma
- k : kecskék száma
- j : juhok száma

- Feltételek:

- $1 \leq s \leq 100$
- $1 \leq k \leq 100$
- $1 \leq j \leq 100$
- Darabszám: $s + k + j = 100$
- Ár: $7/2*s + 4/3*k + 1/2*j = 100$

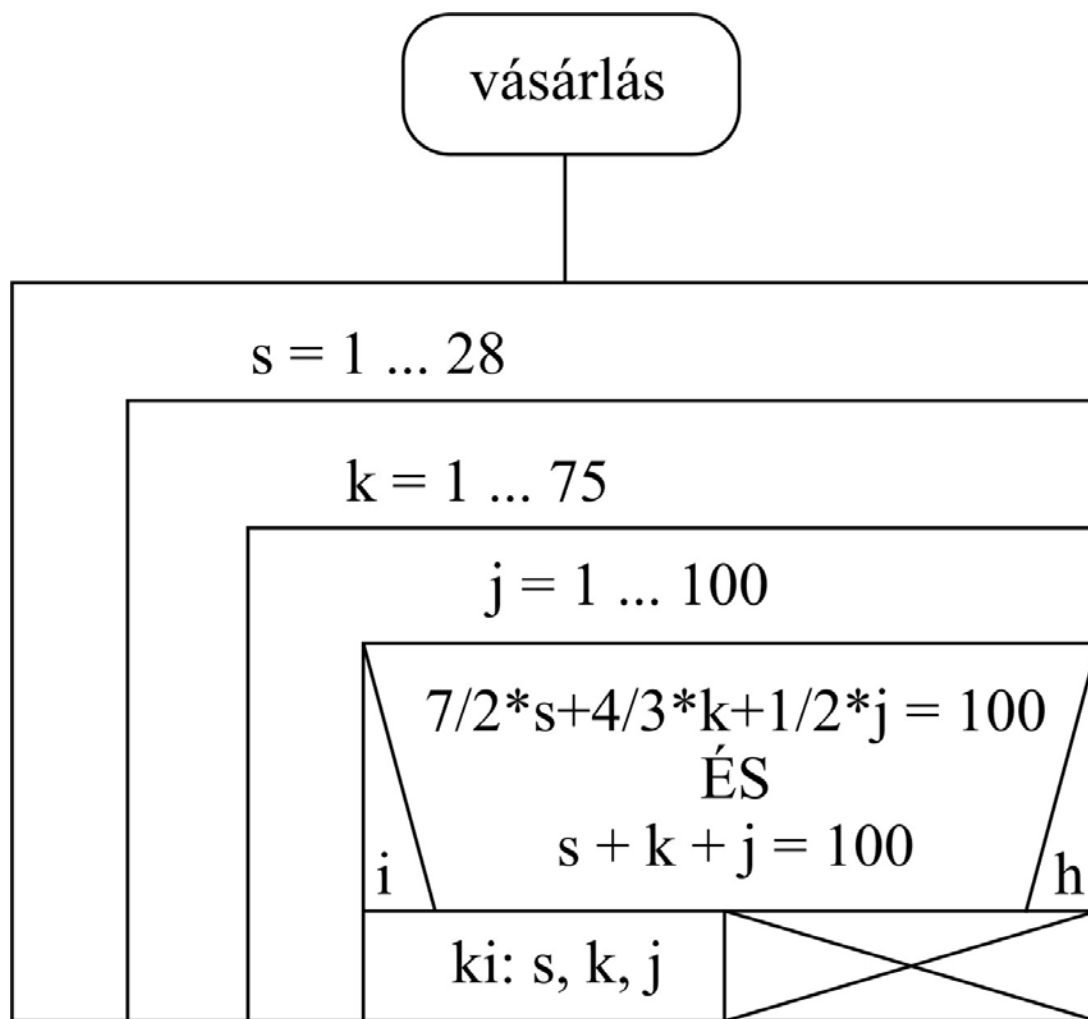
Algoritmus 1



Algoritmus 1, analízis

- Futások száma
 - $100 * 100 * 100 = 1\,000\,000$ $O(n^3)$
- Lehet ennél jobb algoritmust találni?
- Ha csak egyféle állatot venne 100 aranyért akkor:
 - sertés: $7/2 * s = 100$ így 's' egész része: 28
 - kecske: $4/3 * k = 100$ így $k = 75$
 - juh: $1/2 * j = 100$ és bár $j = 200$ lehetne, de csak 100 állatot vett a gazda, így $j = 100$

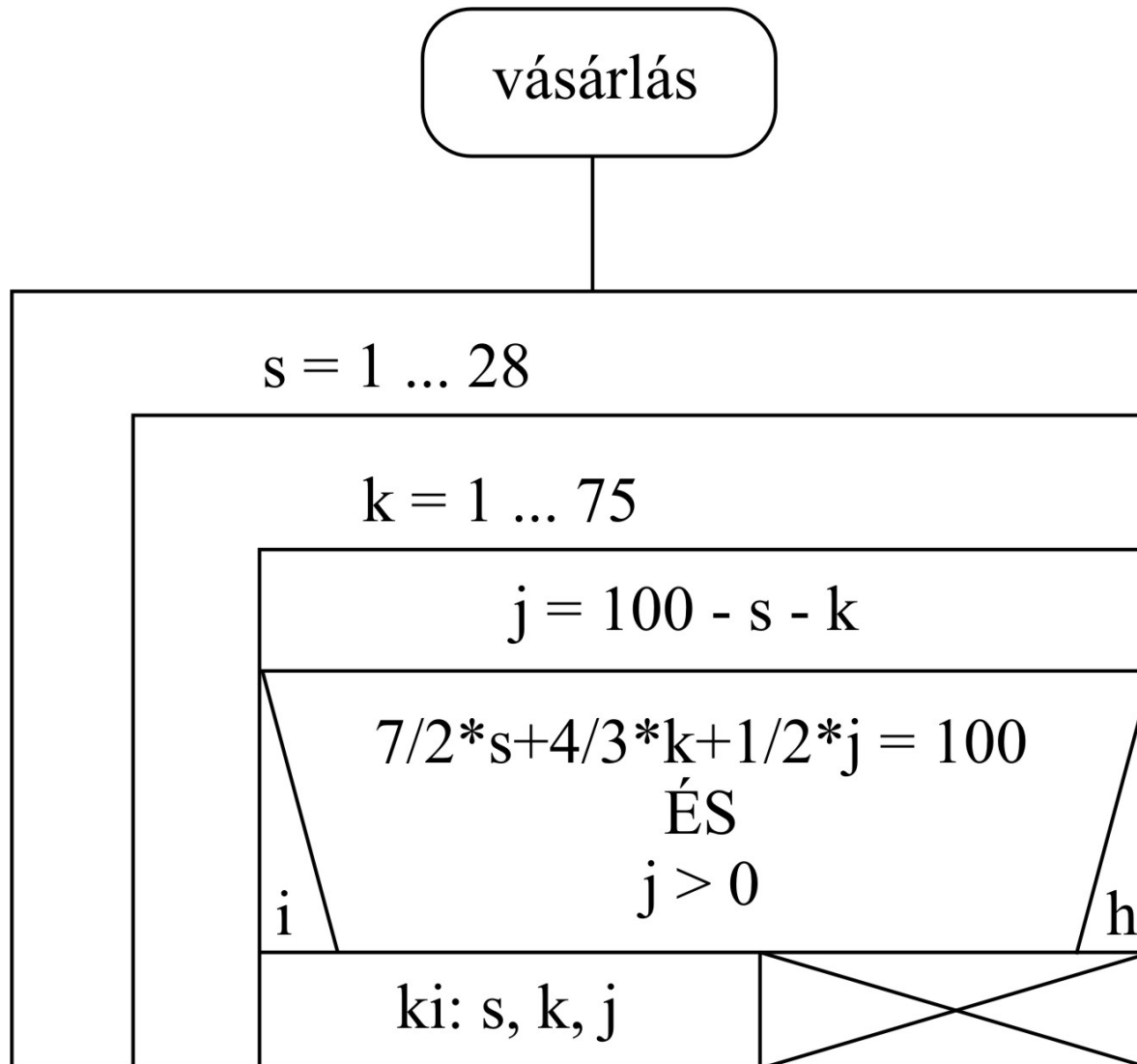
Algoritmus 2



Algoritmus 2, analízis

- $28 * 75 * 100 = 210\,000$
- Lehet ennél jobb algoritmust találni?
- Ha már kétféle állatot kiválasztottunk akkor a harmadik állat száma már adott, így
 - $j = 100 - s - k$
- Következmény:
 - A belső ciklus elhagyható
 - j értéke pontosan meghatározható, de negatív nem lehet, így a feltétel is módosul

Algoritmus 3



Algoritmus 3, analízis

- $28 * 75 = 2100$ $O(n^2)$
- Lehet ennél jobb algoritmust találni?
- Ha az
$$j = 100 - s - k$$
- egyenletet behelyettesítjük az
$$7/2*s + 4/3*k + 1/2*j = 100$$
- egyenletbe, akkor újabb összefüggést kapunk
$$k = 60 - 18/5 * s$$
- Feltétel is módosul: 'k' egész, pozitív kell legyen

Algoritmus 4

vásárlás

$s = 1 \dots 28$

$k = 60 - 18 / 5 * s$

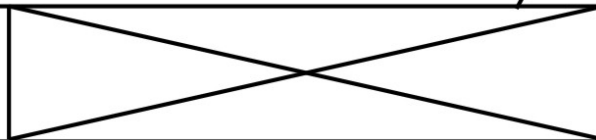
$j = 100 - s - k$

i

k egész ÉS $j > 0$

h

ki: s, k, j



Algoritmus 4, analízis

- 28 lépést kell csak végrehajtanunk!!! $O(n)$
- Lehet ennél jobb algoritmust találni?
- Tudjuk, hogy 'k' pozitív és egész, ez csak akkor lehetséges ha 's' öttenel osztható szám:
 - $s = 5, 10, 15, 20, 25$
- Azonban ha:
 - $s = 25$ akkor $k = -30$ és
 - $s = 20$ akkor $k = -12$ így ezek sem jók

Algoritmus 5

vásárlás

Három lépésben
megoldható a feladat!!!

$O(1)$

$s = 5, 10, 15$

$k = 60 - 18 / 5 * s$

$j = 100 - s - k$

ki: s, k, j

Felhasznált irodalom

- <http://staff.kzs.hu/tamas/programozas/Prgmódsz.htm>
- <http://www.codexonline.hu>
- <http://kognit.edpsy.u-szeged.hu>
- <http://www.prog.hu>
- Bálintné Farkas Judit: Programtervezés, Phare Program HU-94.05, PMMFK