



ELTE | IK

# PROGRAMOZÁS

## Vezérlési szerkezetek

Horváth Győző



# Ismétlés



# Feladatmegoldás lépései

---

- Specifikáció
  - mi a feladat?
  - adatok, megszorítások, összefüggések
- Algoritmus
  - hogyan oldjuk meg a feladatot?
  - milyen lépésekre bontjuk?
  - szekvencia (utasítások egymás után)
  - elágazás (utasítások feltételes végrehajtása)
- Kód
  - megvalósítás a gép számára érthető módon
  - adatok deklarációja, beolvasás, feldolgozás, kiírás

# Összefoglalva – specifikáció

## Feladat:

*Egy ötgyerekes nagycsalád nyaralni indul a Balatonra. Az autópályán a 11 éves, a számok iránt mindig nagy érdeklődést mutató Matyi nem látja a kilométerórát, de szeretné megtudni, mekkora sebességgel mennek. Így elkezd megszámolni hány kilométer táblát hagytak el, és közben az időt is méri. Milyen eredményt kap?*



## Feladat:

*Az út, idő ismeretében határozd meg a sebességet!*



## Specifikáció:

Be:  $s \in \mathbb{R}, t \in \mathbb{R}$

Ki:  $v \in \mathbb{R}$

Ef:  $s \geq 0$  és  $t > 0$

Uf:  $v = s / t$



## Példa:

$s=120; t=1 \rightarrow v=120$

$s=10,5; t=0,1 \rightarrow v=105$

# Összefoglalva – algoritmus

## Specifikáció:

Be:  $s \in \mathbb{R}, t \in \mathbb{R}$   
Ki:  $v \in \mathbb{R}$   
Ef:  $s \geq 0$  és  $t > 0$   
Uf:  $v = s / t$

Változó  
 $s, t: \text{Valós},$   
 $v: \text{Valós}$

Be:	$s, t$	$[s \geq 0 \text{ és } t > 0)$
$v := s/t$		
Ki:	$v$	

$v := s/t$

# Összefoglalva – kód

## Specifikáció:

Be:  $s \in \mathbb{R}, t \in \mathbb{R}$   
Ki:  $v \in \mathbb{R}$   
Ef:  $s \geq 0$  és  $t > 0$   
Uf:  $v = s / t$

Változó  
 $s, t$ : Valós,  
 $v$ : Valós



```
// Deklaráció
double s, t;
double v;
// Beolvasás
Console.Write("s = ");
double.TryParse(Console.ReadLine(), out s);

Console.Write("t = ");
double.TryParse(Console.ReadLine(), out t);

// Feldolgozás
v = s / t;

// Kiírás
Console.WriteLine("A sebesség: {0}", v);
```

**Be:**  $s, t$  [ $s \geq 0$  és  $t > 0$ ]

$v := s / t$

**Ki:**  $v$

# Adatok és vezérlési szerkezetek

---

- Adatok
  - Egyszerű, elemi adatok
  - Specifikáció, algoritmus, kód
- Vezérlési szerkezetek
  - Szekvencia
  - Algoritmus, kód
  - Beolvasás, kiírás, értékadás

# Adatok, típusok, változók

Specifikáció	Algoritmus	Kód
$\mathbb{Z}$	Egész	sbyte, short, <b>int</b> , long
$\mathbb{N}$	Természetes	byte, ushort, <b>uint</b> , ulong
$\mathbb{R}$	Valós	float, <b>double</b>
$\mathbb{L}$	Logikai	bool
$\mathbb{S}$	Szöveg	string
$\mathbb{C}$	Karakter	char

Specifikáció	Algoritmus	Kód
Be: $a \in \mathbb{R}$	a: Valós	<b>double</b> a;



# Megfelelések

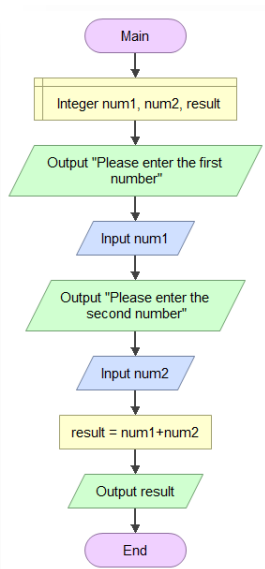
Algoritmus	Kód
<code>:=</code>	<code>=</code>
<code>=</code>	<code>==</code>
<code>és</code>	<code>&amp;&amp;</code>
<code>vagy</code>	<code>  </code>
<code>nem</code>	<code>!</code>
Be: a	<code>Console.Write("a = "); double.TryParse(Console.ReadLine(), out a);</code>
Ki: a	<code>Console.WriteLine(a); Console.WriteLine("a = {0}", a);</code>

# Szekvencia



# Szekvencia

- Hétköznapi algoritmus
  - pl. recept
- Algoritmusleíró nyelvek



húst megtisztítani

1cm vastag szeletekre vágni

klopfolás

...

## Elkészítés

- A húst a zsiradéktól megtisztítjuk, 1 cm vastag szeletekre vágjuk. Enyhén mindkét oldalán kiklopfoljuk, kb. fél centi vastag szeleteket kapunk. Egy csipet sóval mindkét oldalát meghintjük.
1. oldalán kiklopfoljuk, kb. fél centi vastag szeleteket kapunk. Egy csipet sóval mindkét oldalát meghintjük.
  2. Előbb lisztbe, majd kicsi sóval elkevert, felvert tojásba, végül zsemlemorzsába forgatjuk a húsokat.
  3. Közepesen forró olajban, ami ellepi a hússzeleteket, szép aranybarnára kisütjük, először az egyik, majd a másik oldalát. (Én két részletben sütöttem ki: adagonként 30 perc alatt)
  4. Háztartási papírtörülővel bélelt tálba szedjük, hogy a felesleges olajat felitassuk róla.



húst megtisztítani  
1cm vastag szeletekre vágni  
klopfolás

...

# Szekvencia

---

- Utasítások egymás utáni végrehajtása
- Korábbi példa
  - Programok általános felépítése
    - beolvasás
    - feldolgozás
    - kiírás
  - Ez három művelet/alprogram szekvenciája
- Esetek
  - több adat kiszámítása
  - segédadat használata (közbülső adat kiszámítása)

# 1. példa: több adat kiszámítása

Határozd meg egy kétjegyű szám első és második számjegyét!

Példa:  $n=42 \rightarrow$  számjegy1=4, számjegy2=2

## Specifikáció

**Be:**  $n \in \mathbb{N}$

**Ki:** számjegy1  $\in \mathbb{N}$ , számjegy2  $\in \mathbb{N}$

**Ef:**  $n \geq 10$  és  $n \leq 99$  //  $n \in [10..99]$

**Uf:** számjegy1 =  $n \text{ div } 10$  és  
számjegy2 =  $n \text{ mod } 10$

Vagy:

**Uf:** számjegy1 \* 10 + számjegy2 = n

# 1. példa: több adat kiszámítása

Határozd meg egy kétjegyű szám első és második számjegyét!

## Algoritmus

### Specifikáció

**Be:**  $n \in \mathbb{N}$

**Ki:** számjegy1  $\in \mathbb{N}$ , számjegy2  $\in \mathbb{N}$

**Ef:**  $n \geq 10$  és  $n \leq 99$

**Uf:** számjegy1 =  $n \text{ div } 10$  és  
számjegy2 =  $n \text{ mod } 10$

Be:  $n$  [ $n \geq 10$  és  $n \leq 99$ ]

számjegy1 :=  $n \text{ div } 10$

számjegy2 :=  $n \text{ mod } 10$

Ki: számjegy1, számjegy2

## 2. példa: segédadat használata

Határozd meg egy kétjegyű szám első számjegye nagyobb-e, mint a második számjegye!

### Specifikáció

**Be:**  $n \in \mathbb{N}$

**Sa:** számjegy1  $\in \mathbb{N}$ ,  
számjegy2  $\in \mathbb{N}$

**Ki:** nagyobb  $\in \mathbb{L}$

**Ef:**  $n \geq 10$  és  $n \leq 99$

**Uf:** számjegy1 =  $n \text{ div } 10$  és  
számjegy2 =  $n \text{ mod } 10$  és  
nagyobb = számjegy1 > számjegy2

### Specifikáció

**Be:**  $n \in \mathbb{N}$

**Ki:** nagyobb  $\in \mathbb{L}$

**Ef:**  $n \geq 10$  és  $n \leq 99$

**Uf:** nagyobb =  
 $n \text{ div } 10 > n \text{ mod } 10$

Példa:  $n=42 \rightarrow$  nagyobb=igaz

## 2. példa: segédadat használata

Határozd meg egy kétjegyű  
szám első és második  
számjegyét!

### Algoritmus

#### Specifikáció

**Be:**  $n \in \mathbb{N}$

**Sa:** számjegy1  $\in \mathbb{N}$ ,  
számjegy2  $\in \mathbb{N}$

**Ki:** nagyobb  $\in \mathbb{L}$

**Ef:**  $n \geq 10$  és  $n \leq 99$

**Uf:** számjegy1 =  $n \text{ div } 10$  és  
számjegy2 =  $n \text{ mod } 10$  és  
nagyobb = számjegy1 > számjegy2

Local declarations

Változó

n:Egész,

számjegy1:Egész,

számjegy2:Egész,

nagyobb:Logikai

Be:  $n$  [ $n \geq 10$  és  $n \leq 99$ ]

számjegy1 :=  $n \text{ div } 10$

számjegy2 :=  $n \text{ mod } 10$

nagyobb := számjegy1 > számjegy2

Ki: nagyobb



## 2. példa: segédadat használata

Határozd meg egy kétjegyű  
szám első és második  
számjegyét!

### Algoritmus

#### Specifikáció

**Be:**  $n \in \mathbb{N}$

**Sa:** számjegy1  $\in \mathbb{N}$ ,  
számjegy2  $\in \mathbb{N}$

**Ki:** nagyobb  $\in \mathbb{L}$

**Ef:**  $n \geq 10$  és  $n \leq 99$

**Uf:** számjegy1 =  $n \div 10$  és  
számjegy2 =  $n \bmod 10$  és  
nagyobb = számjegy1 > számjegy2

Local declarations

Változó

számjegy1:Egész,  
számjegy2:Egész

**Be:**  $n$  [ $n \geq 10$  és  $n \leq 99$ ]

számjegy1 :=  $n \div 10$

számjegy2 :=  $n \bmod 10$

nagyobb := számjegy > számjegy2

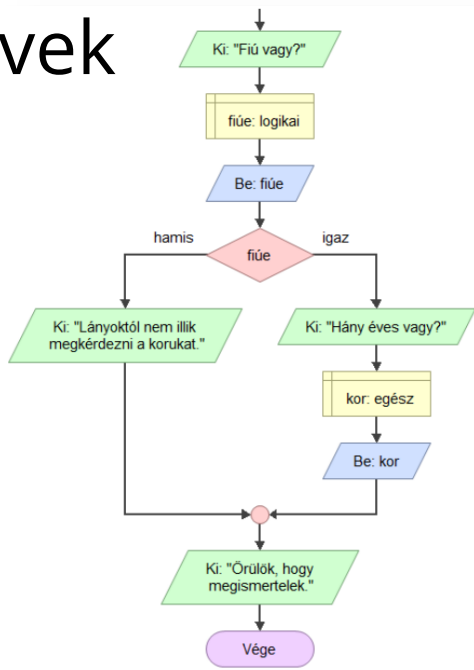
**Ki:** nagyobb

# Elágazás



# Elágazás

- Hétköznapi algoritmus
  - pl: ügyintézés
- Algoritmusleíró nyelvek



Ha feltétel akkor  
utasítások igaz esetén  
különben  
utasítások hamis esetén  
Elágazás vége

T	rendelkezik saját tőkével?	F
😊	bankkal egyeztetni hitelfelvétel	

# Feladatok elágazásra: vércsoport – 1

## Feladat:

*Egy ember vércsoportját (Rh negatív vagy pozitív) egy génpár határozza meg. Mindkét gén lehet „+” vagy „-” típusú. A „++” és a „+-” típusúak az „Rh pozitívok”, a „--” típusúak pedig az „Rh negatívok”.*

*Írj programot, amely megadja egy ember vércsoportját a génpárja ismeretében!*

Példa:  $x=+$ ,  $y=-$   $\rightarrow$   $v=\text{Rh}+$

# Feladatok elágazásra: vércsoport

Példa:  $x="+", y="-" \rightarrow v="Rh+"$

## Specifikáció:

Be:  $x \in C, y \in C$

Ki:  $v \in S$

Ef:  $(x="+" \text{ vagy } x="-") \text{ és } (y="+" \text{ vagy } y="-")$

Uf:  $( (x="+" \text{ vagy } y="+") \text{ és } v="Rh+") \text{ vagy } ( (x="-" \text{ és } y="-") \text{ és } v="Rh-")$

$C$ =Karakterek halmaza

$S$ =Karakter-sorozatok  
(szövegek) halmaza

Ef:  $x, y \in \{ "+", "-" \}$

## Algoritmus:

$\text{nem}(x="+" \text{ vagy } y="+")$

Elhagyjuk a  
változók  
deklarálását, a  
beolvasást és a  
kiírást

T	$x="+" \text{ vagy } y="+"$		F
	$v := "Rh+"$	$v := "Rh-"$	

# Feladatok elágazásra: vércsoport

a	b	a->b
igaz	igaz	igaz
igaz	hamis	hamis
hamis	igaz	igaz
hamis	hamis	igaz

## Specifikáció:

Be:  $x \in C, y \in C$

Ki:  $v \in S$

Ef:  $(x = "+" \text{ vagy } x = "-") \text{ és } (y = "+" \text{ vagy } y = "-")$

Uf:  $( (x = "+" \text{ vagy } y = "+") \rightarrow v = "Rh+" ) \text{ és } ( \text{nem}(x = "+" \text{ vagy } y = "+") \rightarrow v = "Rh-" )$

## Algoritmus:

$x = "+" \text{ vagy } y = "+"$	$\text{nem}(x = "+" \text{ vagy } y = "+")$
$v := "Rh+"$	$v := "Rh-"$

T	$x = "+" \text{ vagy } y = "+"$	F
	$v := "Rh+"$	$v := "Rh-"$

# Feladatok elágazásra: vércsoport – 2

## Feladat:

*Egy ember vércsoportját (A, B, AB vagy 0) egy génpár határozza meg. Mindkét gén lehet **a**, **b** vagy **0** típusú.*

*A vércsoport meghatározása:  $A=\{aa,a0,0a\}$ ;  $B=\{bb,b0,0b\}$ ;  $AB=\{ab,ba\}$ ;  $0=\{00\}$ .*

*Írj programot, amely megadja egy ember vércsoportját a génpárja ismeretében!*

**Példa:**  $x="a", y="b" \rightarrow v="AB"$

# Feladatok elágazásra: vércsoport – 2

**Példa:**  $x="a", y="b" \rightarrow v="AB"$

## Specifikáció:

Be:  $x \in C, y \in C$

Ki:  $v \in S$

Ef:  $(x="a" \text{ vagy } x="b" \text{ vagy } x="0") \text{ és } (y="a" \text{ vagy } y="b" \text{ vagy } y="0")$

Uf:  $((x="a" \text{ és } y \neq "b" \text{ vagy } x \neq "b" \text{ és } y="a") \rightarrow v="A") \text{ és } ((x="b" \text{ és } y \neq "a" \text{ vagy } x \neq "a" \text{ és } y="b") \rightarrow v="B") \text{ és } ((x="a" \text{ és } y="b" \text{ vagy } x="b" \text{ és } y="a") \rightarrow v="AB") \text{ és } ((x="0" \text{ és } y="0") \rightarrow v="0")$

Egy ember vércsoportját (A, B, AB vagy 0) egy génpár határozza meg. Mindkét gén lehet **a**, **b** vagy **0** típusú.

A vércsoport meghatározása:  $A=\{aa,a0,0a\}$ ;  $B=\{bb,b0,0b\}$ ;  $AB=\{ab,ba\}$ ;  $0=\{00\}$ .

Írj programot, amely megadja egy ember vércsoportját a génpárja ismeretében!



# Feladatok elágazásra: vércsoport – 2

## Algoritmus<sub>2</sub>:

Sokirányú  
elágazással.

$x = "a"$ és $y \neq "b"$ vagy $x \neq "b"$ és $y = "a"$	$x = "b"$ és $y \neq "a"$ vagy $x \neq "a"$ és $y = "b"$	$x = "a"$ és $y = "b"$ vagy $x = "b"$ és $y = "a"$	$x = "0"$ és $y = "0"$
$v := "A"$	$v := "B"$	$v := "AB"$	$v := "0"$

### Specifikáció:

Be:  $x \in C, y \in C$

Ki:  $v \in S$

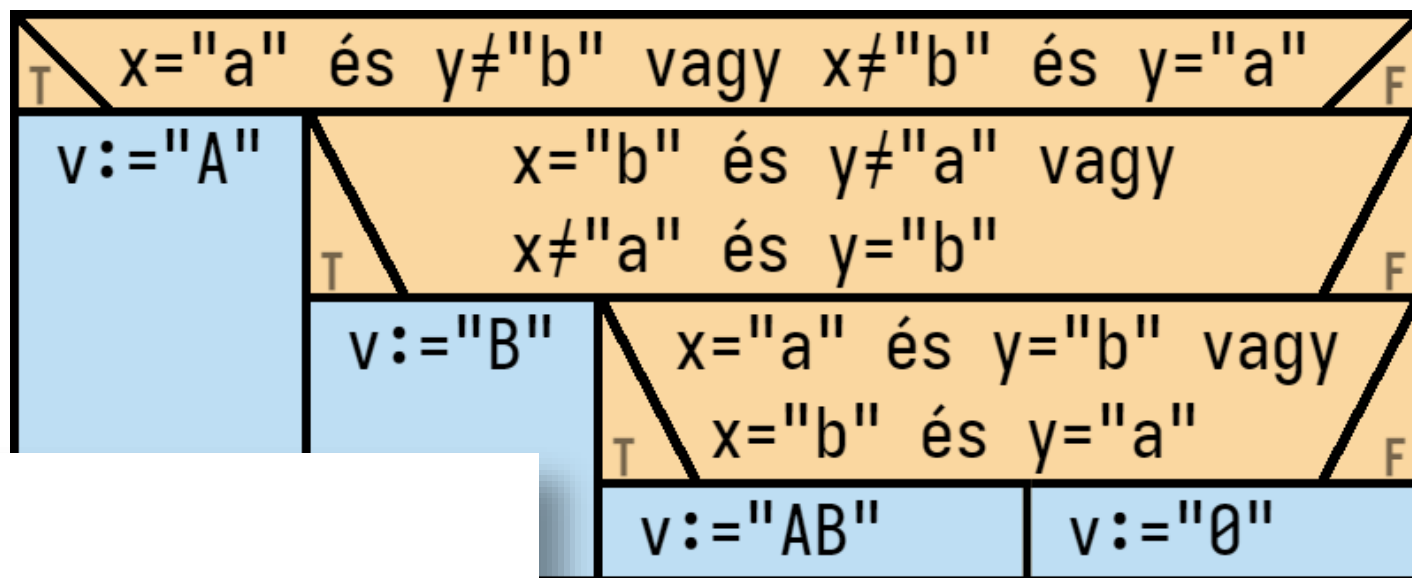
Ef:  $(x = "a" \text{ vagy } x = "b" \text{ vagy } x = "0")$  és  
 $(y = "a" \text{ vagy } y = "b" \text{ vagy } y = "0")$

Uf:  $((x = "a" \text{ és } y \neq "b" \text{ vagy } x \neq "b" \text{ és } y = "a") \rightarrow v = "A")$  és  
 $((x = "b" \text{ és } y \neq "a" \text{ vagy } x \neq "a" \text{ és } y = "b") \rightarrow v = "B")$  és  
 $((x = "a" \text{ és } y = "b" \text{ vagy } x = "b" \text{ és } y = "a") \rightarrow v = "AB")$  és  
 $((x = "0" \text{ és } y = "0") \rightarrow v = "0")$

# Feladatok elágazásra: vércsoport – 2

## Algoritmus<sub>1</sub>:

Kétirányú  
elágazások  
egymásba  
ágyazásával.



### Specifikáció:

Be:  $x \in C$ ,  $y \in C$

Ki:  $v \in S$

Ef:  $(x = "a" \text{ vagy } x = "b" \text{ vagy } x = "0")$  és  
 $(y = "a" \text{ vagy } y = "b" \text{ vagy } y = "0")$

Uf:  $(x = "a" \text{ és } y \neq "b" \text{ vagy } x \neq "b" \text{ és } y = "a" \rightarrow v = "A")$  és  
 $(x = "b" \text{ és } y \neq "a" \text{ vagy } x \neq "a" \text{ és } y = "b" \rightarrow v = "B")$  és  
 $(x = "a" \text{ és } y = "b" \text{ vagy } x = "b" \text{ és } y = "a" \rightarrow v = "AB")$  és  
 $(x = "0" \text{ és } y = "0" \rightarrow v = "0")$

# Feladatok elágazásra: vércsoport – 2

## Algoritmus<sub>3</sub>:

Segédváltozók  
bevezetésével.

### Specifikáció:

Be:  $x \in C$ ,  $y \in C$

Ki:  $v \in S$

Ef:  $(x = "a" \text{ vagy } x = "b" \text{ vagy } x = "\emptyset") \text{ és } (y = "a" \text{ vagy } y = "b" \text{ vagy } y = "\emptyset")$

Uf:  $(x = "a" \text{ és } y \neq "b" \text{ vagy } x \neq "b" \text{ és } y = "a" \rightarrow v = "A") \text{ és } (x = "b" \text{ és } y \neq "a" \text{ vagy } x \neq "a" \text{ és } y = "b" \rightarrow v = "B") \text{ és } (x = "a" \text{ és } y = "b" \text{ vagy } x = "b" \text{ és } y = "a" \rightarrow v = "AB") \text{ és } (x = "\emptyset" \text{ és } y = "\emptyset" \rightarrow v = "\emptyset")$

Local declarations

Változó

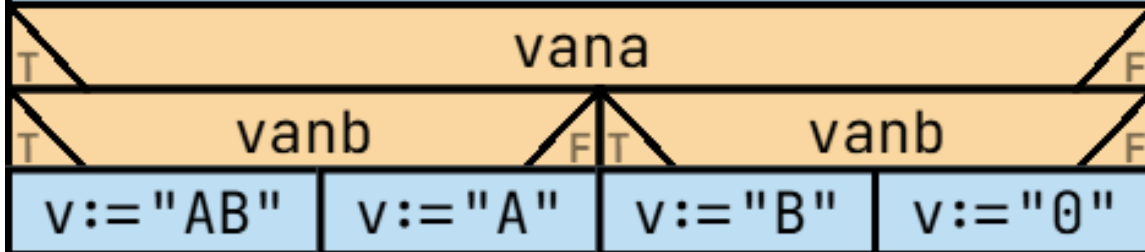
vana,

vanb: Logikai

Lokális változók  
deklarálása

$vana := x = "a" \text{ vagy } y = "a"$

$vanb := x = "b" \text{ vagy } y = "b"$



# Elágazás

## Kód:

kétirányú

```
if (felt) {  
    utasítás1  
}  
else {  
    utasítás2  
}
```

elágazás

sokirányú  
(általános)

```
if (felt1) {  
    utasítás1  
}  
else if (...) {  
    ...  
}  
else if (feltn) {  
    utasításn  
}  
else {  
    utasítás  
}
```

# Elágazás

## Kód:

sokirányú  
elágazás (speciális)

```
switch (kif)
{
    case érték1: utasítás1; break;
    case ... : ... ; break;
    case értékN: utasításn; break;
    default utasítás elhagyható ; break;
}
```

# Kód

$x = "a"$ és $y \neq "b"$ vagy $x \neq "b"$ és $y = "a"$	$x = "b"$ és $y \neq "a"$ vagy $x \neq "a"$ és $y = "b"$	$x = "a"$ és $y = "b"$ vagy $x = "b"$ és $y = "a"$	$x = "0"$ és $y = "0"$
$v := "A"$	$v := "B"$	$v := "AB"$	$v := "0"$

```
// 1. deklarálás
char x, y;
string v = "";
// 2. beolvasás
Console.Write("x = ");
char.TryParse(Console.ReadLine(), out x);
Console.Write("y = ");
char.TryParse(Console.ReadLine(), out y);
// 3. feldolgozás
if ((x == 'a' && y != 'b') || (x != 'b' && y == 'a')) {
    v = "A";
}
else if ((x == 'b' && y != 'a') || (x != 'a' && y == 'b')) {
    v = "B";
}
else if ((x == 'a' && y == 'b') || (x == 'b' && y == 'a')) {
    v = "AB";
}
else if (x == '0' && y == '0') {
    v = "0";
}
// 4. kiírás
Console.WriteLine("v = {0}", v);
```

# Háromszög

---

## Feladat:

*3 szám lehet-e egy derékszögű háromszög 3 oldala?*

## Megoldás:

- Akkor lehet, ha  $a^2 + b^2 = c^2$
- Akkor nem lehet, ha ez nem teljesül

# Példa: háromszög

Tessék kipróbálni az alábbi adatokkal!

a: 3

b: 4

c: 5.0000000000000001

lehet: false

## Feladat:

*3 szám lehet-e egy derékszögű háromszög 3 oldala?*

## Specifikáció:

Be:  $a \in \mathbb{R}$ ,  $b \in \mathbb{R}$ ,  $c \in \mathbb{R}$

$\mathbb{R}$ =Valós számok **halmaza**

Ki: lehet  $\in \mathbb{L}$

$\mathbb{L}$ =Logikai értékek **halmaza**

Ef:  $a > 0$  és  $b > 0$  és  $c > 0$

Uf:  $(a^2 + b^2 = c^2$  és  $\text{lehet} = \text{igaz})$  vagy  
 $(a^2 + b^2 \neq c^2$  és  $\text{lehet} = \text{hamis})$

Megjegyzés: a 3 szám sorrendjét ezek szerint implicite rögzítettük  
– c az átfogó hossza!



# Példa: háromszög

---

## Feladat:

*3 szám lehet-e egy derékszögű háromszög 3 oldala?*

## Specifikáció<sub>2</sub>:

Be:  $a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$

Ki: lehet  $\in \mathbb{L}$

Ef:  $0 < a$  és  $a \leq b$  és  $b \leq c$

Uf:  $(a^2 + b^2 = c^2$  és lehet = igaz) vagy  
 $(a^2 + b^2 \neq c^2$  és lehet = hamis)

Megjegyzés: a 3 szám sorrendjét ezek szerint **explicit**e rögzítettük  
– c az átfogó hossza!

# Példa: háromszög

## Implikáció:

a „ha-akkor” logikai kifejezése

a	b	a→b
igaz	igaz	igaz
igaz	hamis	hamis
hamis	igaz	igaz
hamis	hamis	igaz

Uf:  $(a * a + b * b = c * c \text{ és lehet=igaz}) \text{ vagy}$   
 $(a * a + b * b \neq c * c \text{ és lehet=hamis})$

Uf:  $(a * a + b * b = c * c \rightarrow \text{lehet=igaz}) \text{ és}$   
 $(a * a + b * b \neq c * c \rightarrow \text{lehet=hamis})$

# Példa: háromszög

A valós típusú azonosság körül adódhatnak problémák a valós típusú adatok ábrázolása miatt.  
A precíz megoldás:  
 $|a^2+b^2-c^2| < \text{epszilon}$

Be:  $a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$

Ki:  $\text{lehet} \in \mathbb{L}$

Ef:  $0 < a$  és  $a \leq b$  és  $b \leq c$

Uf:  $(a^2+b^2=c^2 \rightarrow \text{lehet}=\text{igaz})$  és  
 $(a^2+b^2 \neq c^2 \rightarrow \text{lehet}=\text{hamis})$

**Algoritmus:**

T	$a^2+b^2=c^2$	F
lehet:=igaz	lehet:=hamis	

# Kód megoldás

```
// Deklaráció
double a, b, c;
bool lehet;
// Beolvasás
Console.Write("a = ");
double.TryParse(Console.ReadLine(), out a);

Console.Write("b = ");
double.TryParse(Console.ReadLine(), out b);

Console.Write("c = ");
double.TryParse(Console.ReadLine(), out c);

// Feldolgozás
if (a * a + b * b == c * c) {
    lehet = true;
}
else {
    lehet = false;
}

// Kiírás
Console.WriteLine("Lehet derékszögű háromszög: {0}", lehet);
```

A valós típusú azonosság körül adódhatnak problémák a valós típusú adatok ábrázolása miatt.

A precíz megoldás:  
 $Abs(a*a+b*b-c*c) < \epsilon$

a*a+b*b=c*c	
T	F
lehet:=igaz	lehet:=hamis

# Példa: háromszög

---

## Specifikáció<sub>3</sub>:

Be:  $a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$

Ki:  $lehet \in \mathbb{L}$

Ef:  $0 < a$  és  $a \leq b$  és  $b \leq c$

Uf:  $lehet = (a*a + b*b = c*c)$

## Algoritmus:

$lehet := (a*a + b*b = c*c)$

# Példa: háromszög

Egy másik **algorithmus** a lényegi részre:

Local declarations
<b>Változó</b>
aa,bb,cc:Valós
aa:=a*a
bb:=b*b
cc:=c*c
lehet:=(aa+bb=cc)

Segédváltozók  
deklarálása

Bevezethetők/-endők segéd (belső, saját) változók.

# Kód megoldás

---

```
// Deklaráció
double a, b, c;
bool lehet;
// Beolvasás
Console.Write("a = ");
double.TryParse(Console.ReadLine(), out a);

Console.Write("b = ");
double.TryParse(Console.ReadLine(), out b);

Console.Write("c = ");
double.TryParse(Console.ReadLine(), out c);

// Feldolgozás
lehet = (a * a + b * b == c * c);

// Kiírás
Console.WriteLine("Lehet derékszögű háromszög: {0}", lehet);
```

# Kód megoldás

---

```
// Deklaráció és beolvasás ld. korábban
// Előfeltétel ellenőrzés
if (0 < a && a <= b && b <= c) {
    // Feldolgozás
    lehet = (a * a + b * b == c * c);

    // Kiírás
    if (lehet) {
        Console.WriteLine("Lehet derékszögű háromszög");
    }
    else {
        Console.WriteLine("Nem lehet derékszögű háromszög");
    }
}
else {
    Console.WriteLine("Nem megfelelő értékek! Futtassa újra!");
}
```



# Másodfokú egyenlet

**Feladat:** Adjuk meg a másodfokú egyenlet egy megoldását!

Az egyenlet:  $ax^2+bx+c=0$  .

**Specifikáció<sub>1</sub>:**

Be:  $a \in \mathbb{R}$ ,  $b \in \mathbb{R}$ ,  $c \in \mathbb{R}$

Ki:  $x \in \mathbb{R}$

Ef: -

Uf:  $a*x*x+b*x+c=0$  //  $ax^2+bx+c=0$

Megjegyzés: az uf. nem ad algoritmizálható információt. Nem baj, sőt tipikus, de ... próbálkozzunk még!

Megoldóképlet:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4 * a * c}}{2 * a}$$

# Másodfokú egyenlet

---

## Specifikáció<sub>2</sub>:

Be:  $a \in \mathbb{R}, b \in \mathbb{R}, c \in \mathbb{R}$

Ki:  $x \in \mathbb{R}$

Ef:  $a \neq 0$

Uf:  $x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$

## Nyitott kérdések:

- *Mindig/Mikor van megoldás?*
- *Egy megoldás van?*

# Másodfokú egyenlet

---

## Specifikáció:

Be:  $a \in \mathbb{R}$ ,  $b \in \mathbb{R}$ ,  $c \in \mathbb{R}$

Ki:  $x \in \mathbb{R}$ ,  $van \in L$

Ef:  $a \neq 0$

Uf:  $van = (b*b - 4*a*c \geq 0)$  és

$van \rightarrow x = \frac{-b + \sqrt{b*b - 4*a*c}}{2*a}$

# Másodfokú egyenlet

---

## Kimenetbővítés:

Ki:  $x \in \mathbb{R}$ ,  $van \in L$

Uf:  $van = (b^2 - 4ac \geq 0)$  és

$van \rightarrow x = (-b \pm \sqrt{b^2 - 4ac}) / (2a)$

## Nyitott kérdés:

- *Egy megoldás van? – hf.*

# Másodfokú egyenlet

## Algoritmus:

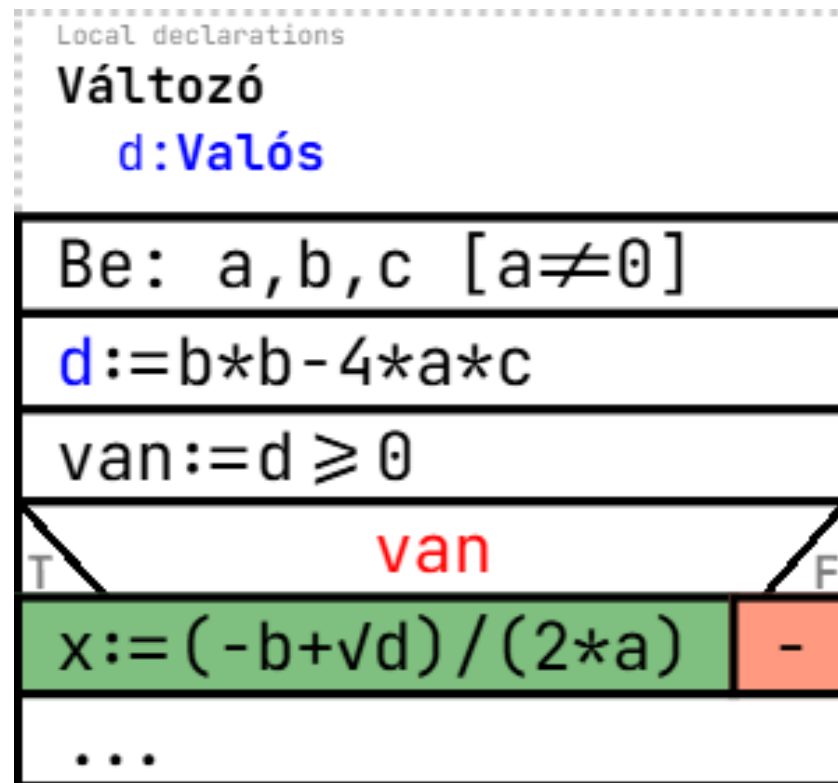
### Specifikáció:

Be:  $a \in \mathbb{R}$ ,  $b \in \mathbb{R}$ ,  $c \in \mathbb{R}$

Ki:  $x \in \mathbb{R}$ ,  $van \in \mathbb{L}$

Ef:  $a \neq 0$

Uf:  $van = (b*b - 4*a*c \geq 0)$  és  
 $van \rightarrow x = (-b + \sqrt{b*b - 4*a*c}) / (2*a)$



# Ciklus



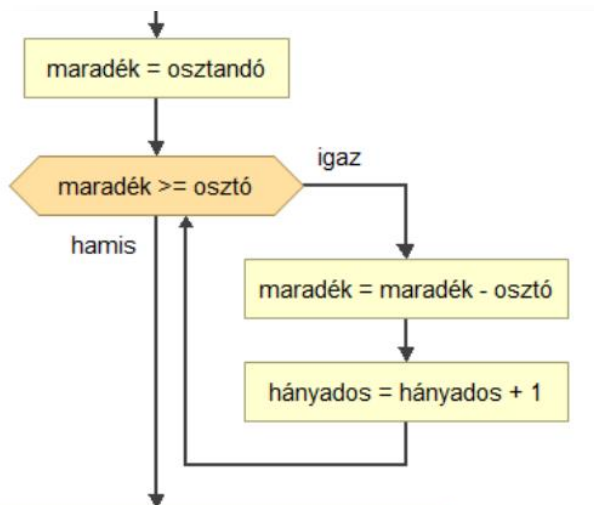
# Ciklus

- Ismételt végrehajtás
- Hétköznapi algoritmusok
  - Addig jár a korsó a kútra, **amíg** el nem törik
  - Recept
- Algoritmusleíró nyelvek

Egyszerűen csak várjuk meg, **amíg** felforr a víz és tegyük bele a tojásokat, így biztos nem okoz majd nagy nehézséget a hámozás.

ÁPRILY LAJOS: ÁMULNI MÉG... (részlet)  
„Ámulni még, **ameddig** lehet, **amíg** a szíved jó ütemre dobban, megőrizni a táguló szemet, mellyel csodálkoztál gyermekkorodban..

Ciklus **amíg** nem törött?  
menj a kútra korsó!  
Ciklus vége



nem törött?

menj a kútra korsó!

# Ciklusok – elemi feladatok

---

## Feladat:

Írd ki 1-től kezdve az első 5 számot!

## Megoldás szekvenciával:

Ki: 1
Ki: 2
Ki: 3
Ki: 4
Ki: 5

Mit tennénk, ha az első 1000 számot kellene kiírni?



# Ciklusok – elemi feladatok

## Feladat:

Írd ki 1-től kezdve az első 5 számot!

## Megoldás számlálós ciklussal:

```
Local declarations  
Változó i:Egész  
i=1..5  
Ki: i
```

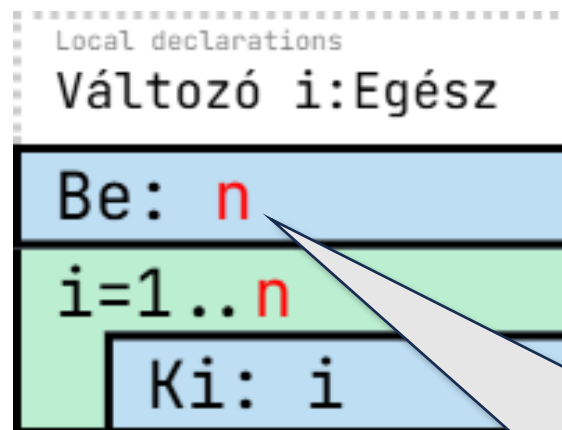
A kiírás számossága könnyen változtatható, a program nem lesz bonyolultabb!

# Ciklusok – elemi feladatok

## Feladat:

Írd ki 1-től kezdve az első  $n$  számot!

## Megoldás számlálós ciklussal:



A kiírás számossága könnyen változtatható, a program nem lesz bonyolultabb!

# Ciklusok – elemi feladatok

---

## Feladat:

Add össze 1-től kezdve az első 5 számot!

## Specifikáció:

**Be:** -

**Ki:**  $\text{összeg} \in \mathbb{N}$

**Ef:** -

**Uf:**  $\text{összeg} = 1 + 2 + 3 + 4 + 5$

## Algoritmus:

$\text{összeg} := 1 + 2 + 3 + 4 + 5$

# Ciklusok – elemi feladatok

## Feladat:

Add össze 1-től kezdve az első  $n$  számot!

## Specifikáció:

**Be:**  $n \in \mathbb{N}$

**Ki:**  $\text{összeg} \in \mathbb{N}$

**Ef:** -

**Uf:**  $\text{összeg} = 1 + 2 + 3 + \dots + n$

## Helyette:

**Uf:**  $\text{összeg} = \text{SZUMMA}(i = 1..n, i)$

Ez nem működik!  
Viszont matematikában erre van  
operátor:

$$\text{összeg} = \sum_{i=1}^n i$$

Local declarations

Változó

$i$ : Egész

$\text{összeg} := 0$

$i = 1..n$

$\text{összeg} := \text{összeg} + i$

# Ciklusok – elemi feladatok

## Feladat:

Add össze 1-től kezdve az első  $n$  számot!

## Specifikáció:

**Uf:**  $\text{összeg} = \text{SZUMMA}(i=1..n, i)$   
 $= 0 + 1 + 2 + 3 + \dots + n$   
 $= (((((0) + 1) + 2) + 3) + \dots + n)$

Mi ismétlődik? Egy zárójeleshez értékhez (részösszeg) való hozzáadás.

## Algoritmus:

```
Local declarations
Változó
i:Egész

összeg := 0
i = 1 .. n
    összeg := összeg + i
```

# Egy szám faktoriálisa

## Feladat:

Határozd egy természetes szám faktoriálisának értékét!

## Specifikáció:

**Be:**  $n \in \mathbb{N}$

**Ki:**  $\text{fakt} \in \mathbb{N}$

**Ef:**  $n > 0$

**Uf:**  $\text{fakt} = n! = 1 * 2 * 3 * \dots * n$

Így is lehet írni:  $n$  eleme  $\mathbb{N}$

Ha van (köz)ismert matematikai művelet / fogalom, lehet használni.

$$\text{fakt} = \prod_{i=1}^{\text{szám}} i$$

## Algoritmus:

Local declarations

Változó  $i$ : Egész

$\text{fakt} := 1$

$i = 2 \dots n$

$\text{fakt} := \text{fakt} * i$

# Ciklusok

## Feladat:

Add meg egy természetes szám ( $>1$ ) **1-től különböző legkisebb osztóját!**

## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $1 < o \leq n$  és  $o \mid n$  és  $\forall i \in [2..o-1]: (i \nmid n)$

Példa:  $n=15 \rightarrow o=3$

## A megoldás reprezentálása:

### Specifikáció:

Be:  $n \in \mathbb{N}$   
Ki:  $o \in \mathbb{N}$   
Ef:  $n > 1$   
Uf:  $1 < o \leq n$  és  $o \mid n$  és  
 $\forall i \in [2..o-1]: (i \nmid n)$

### Változó

$n$ : **Egész**

$o$ : **Egész**

Programváltozók  
deklarálása

Reprezentációs „szabály” a specifikáció  $\rightarrow$  reprezentáció  
áttéréskor:

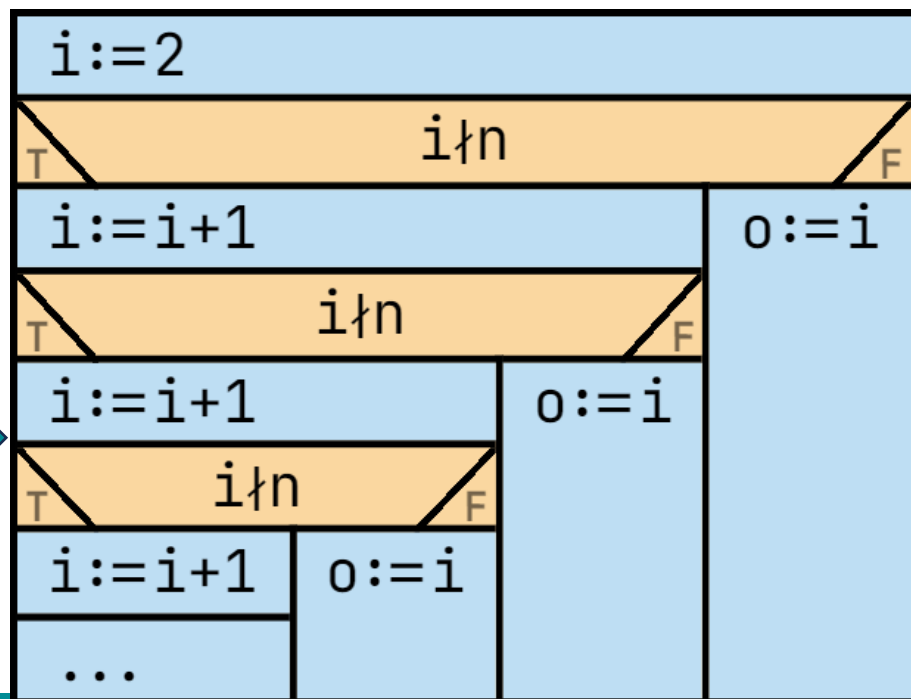
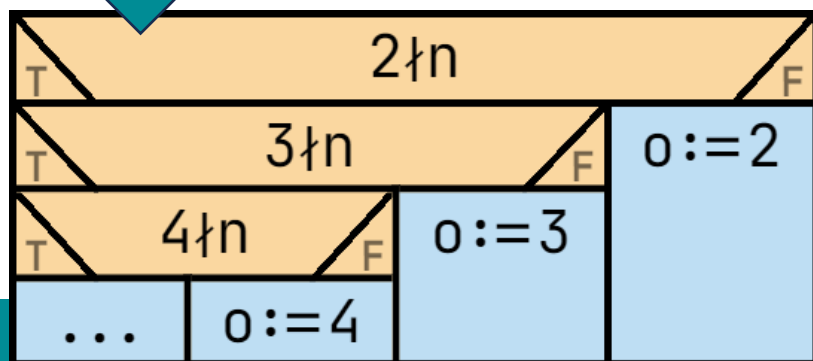
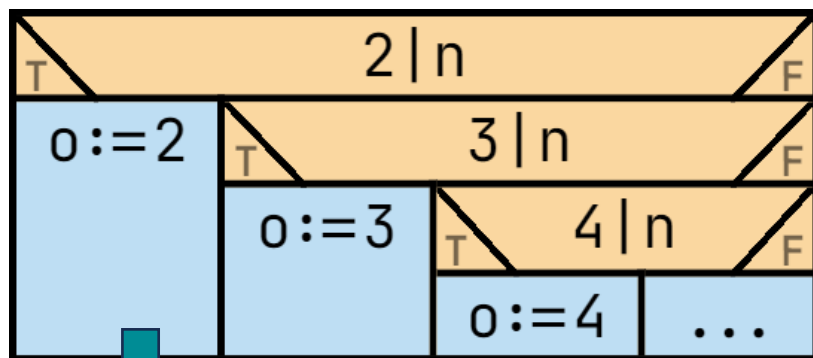
$n \rightarrow$  **Egész**



# Ciklusok

## A megoldás ötlete:

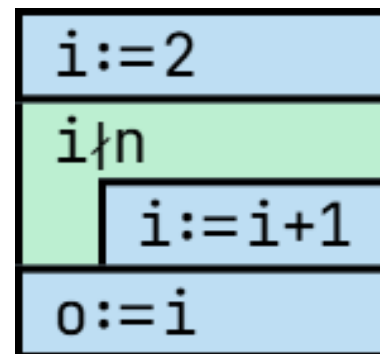
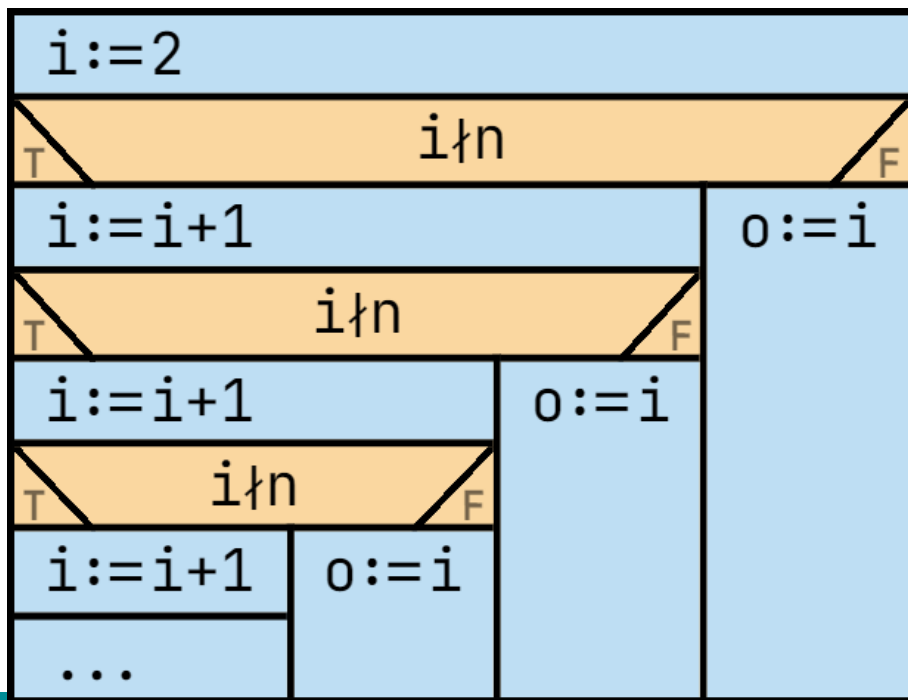
Próbáljuk ki a 2-t; ha nem jó, akkor a 3-at, ha az sem, akkor a 4-et, ...; legkésőbb az  $n$  jó lesz!



# Ciklusok

## A megoldás ötlete:

Próbáljuk ki a 2-t; ha nem jó, akkor a 3-at, ha az sem, akkor a 4-et, ...; legkésőbb az  $n$  jó lesz!



# Ciklusok

## A megoldás ötlete:

Próbáljuk ki a 2-t; ha nem jó, akkor a 3-at,  
ha az sem, akkor a 4-et, ...; legkésőbb az  $n$  jó lesz!

## Az ezt kifejező lényegi algoritmus:

Az  $i$  változó szerepe: végigmenni egy halmaz elemein.

### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $1 < o \leq n$  és  $o | n$  és

$\forall i \in [2..o-1]: (i \nmid n)$

Local declarations

Változó  $i$ : Egész

$i := 2$

$i \nmid n$

$i := i + 1$

$o := i$

Lokális változó  
deklarálása

# Ciklusok

Példa:  $n=15 \rightarrow lko=3; lno=5$

## Feladat:

Határozzuk meg egy természetes szám ( $n>1$ ) **1-től különböző legkisebb** és **önmagától különböző legnagyobb** osztóját!

## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $lko \in \mathbb{N}, lno \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $1 < lko \leq n$  és  $1 \leq lno < n$  és

$lko \mid n$  és  $\forall i \in [2..lko-1]: (i \nmid n)$  és

$lno \mid n$  és  $\forall i \in [lno+1..n-1]: (i \nmid n)$

### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $1 < o \leq n$  és  $o \mid n$  és  
 $\forall i \in [2..o-1]: (i \nmid n)$

# Ciklusok

## Algoritmus:

### Feladat:

Határozzuk meg egy természetes szám ( $n > 1$ ) 1-től különböző legkisebb és önmagától különböző legnagyobb osztóját!

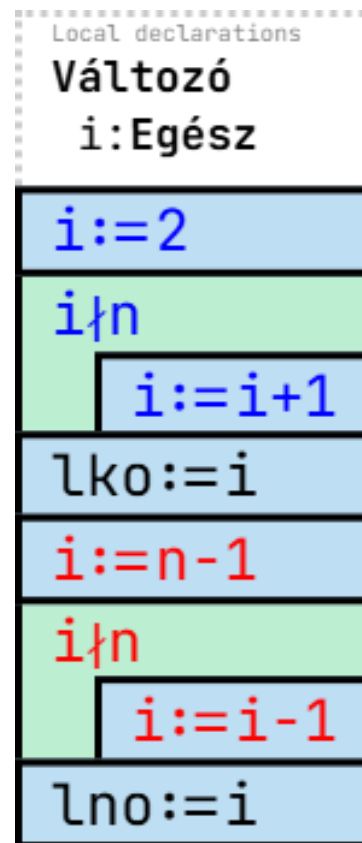
### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $lko \in \mathbb{N}$ ,  $lno \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $1 < lko \leq n$  és  $1 \leq lno < n$  és  
 $lko \mid n$  és  $\forall i \in [2..lko-1]: (i \nmid n)$  és  
 $lno \mid n$  és  $\forall i \in [lno+1..n-1]: (i \nmid n)$



# Ciklusok

## Megjegyzés:

A specifikációból az algoritmus megkapható, de az lno az utófeltételben az lko ismeretében másképp is megfogalmazható:  $lko * lno = n$ !

## Az erre építő algoritmus:

### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $lko \in \mathbb{N}, lno \in \mathbb{N}$

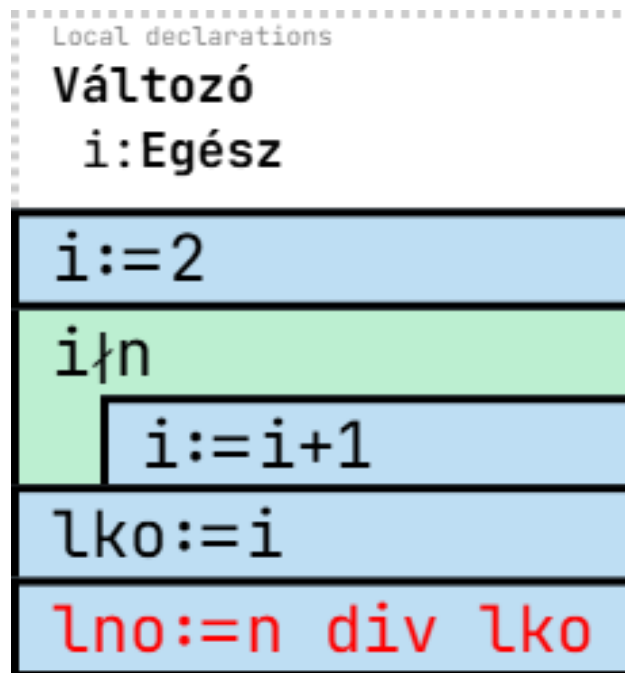
Ef:  $n > 1$

Uf:  $1 < lko \leq n$  és

$lko \mid n$  és

$\forall i \in [2..lko-1]: (i \nmid n)$  és

$lko * lno = n$



# Ciklusok

## Feladat:

Határozzuk meg egy természetes szám ( $n > 1$ ) 1-től és önmagától különböző legkisebb osztóját (ha van)!

## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$ ,  $van \in \mathbb{L}$

Ef:  $n > 1$

Uf:  $van = \exists i \in [2..n-1] : (i | n) \text{ és } van \rightarrow (2 \leq o < n \text{ és } o | n \text{ és } \forall i \in [2..o-1] : (i \nmid n))$

### Példa:

$n=15 \rightarrow van=igaz; o=3$

$n=17 \rightarrow van=hamis; o=???$

# Ciklusok

## Algoritmus:

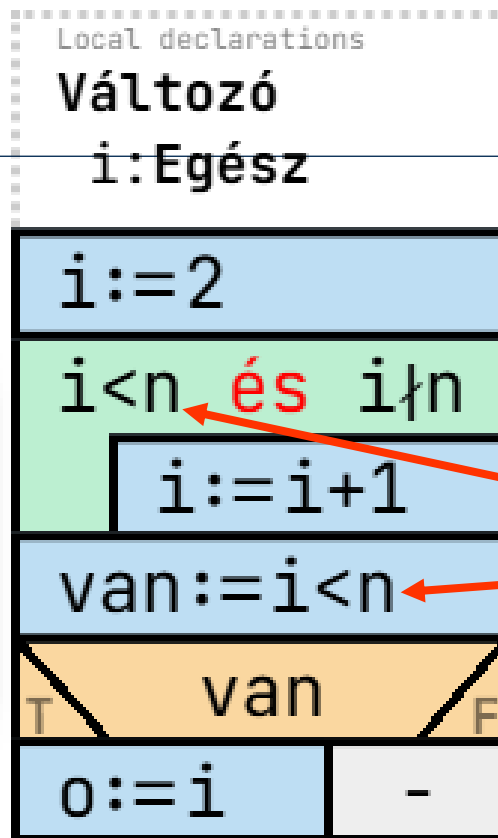
### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$ ,  $van \in \mathbb{L}$

Ef:  $n > 1$

Uf:  $van = \exists i \in [2..n-1] : (i \mid n)$  és  
 $van \rightarrow (2 \leq o < n \text{ és } o \mid n \text{ és } \forall i \in [2..o-1] : (i \nmid n))$



$$i \leq \sqrt{N}$$

$$\begin{aligned} 2 \rightarrow i \leq N \text{ Div } i \leftarrow N \text{ Div } 2 \\ \text{azaz} \\ i * i \leq N \\ \text{azaz} \\ i \leq \sqrt{N} \end{aligned}$$

## Megjegyzés:

Ha  $i$  osztója  $n$ -nek, akkor  $(n \text{ div } i)$  is osztója, azaz elég az osztókat a szám gyökéig keresni!



# Ciklusok

## Feladat:

Határozzuk meg egy természetes szám ( $N > 1$ ) osztói összegét!

## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $s \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $s = \text{SZUM}(i=1..n, i, i | n)$

$$s = \sum_{\substack{i=1 \\ i|n}}^n i$$

A feltételes szumma értelmezéséhez egy példa:

$N=15 \rightarrow \Sigma =$

$i=1 : (1 | 15) \rightarrow 1$

$i=2 : (2 \nmid 15) \rightarrow 1$

$i=3 : (3 | 15) \rightarrow 1+3$

$i=4 : (4 \nmid 15) \rightarrow 1+3$

...

$i=15 : (15 | 15) \rightarrow 1+3+\dots+15$

# Ciklusok

## Algoritmus:

### Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $s \in \mathbb{N}$

Ef:  $n > 1$

Uf:  $s = \text{SZUM}(i=1..n, i, i | n)$

Local declarations

Változó

$i$ : Egész

$s := 0$

$i = 1..n$

$i | n$

$s := s + i$

F

-

Az  $s$  változót nem egy képlettel számoljuk, hanem gyűjtjük benne az eredményt.

Kérdés:

Lehetne itt is  $\sqrt{n}$ -ig menni?

Az  $s := s + i + (n \text{ div } i)$  értékadással?

# Ciklusok

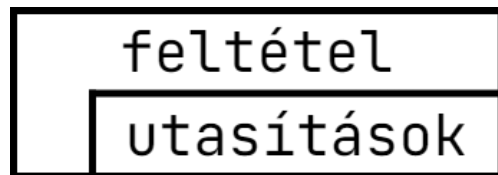
---

## Tanulságok:

- Ha az utófeltételben  $\exists$ ,  $\forall$ , vagy  $\Sigma$  jel van, akkor a megoldás mindig **ciklus**!
- Ha az utófeltételben  $\exists$  vagy  $\forall$  jel van, akkor a megoldás sokszor **feltételes ciklus**!
- Ha az utófeltételben  $\Sigma$  jel van, akkor a megoldás sokszor **számlálós ciklus**! ( $\Pi$  is...)
- **Feltételes  $\Sigma$  esetén a ciklusban elágazás lesz.**

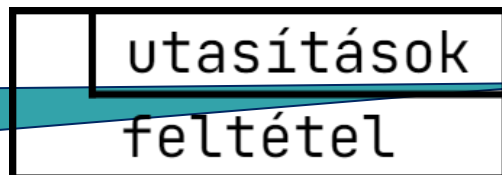
# Ciklusok

## Feltételes ciklus:



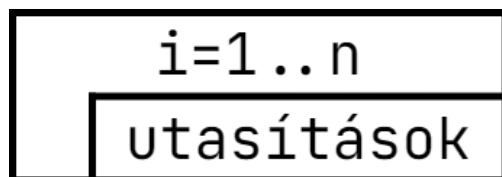
```
while (feltétel){  
    utasítások  
}
```

Tipikus előfordulás: a beolvasás ellenőrzésénél

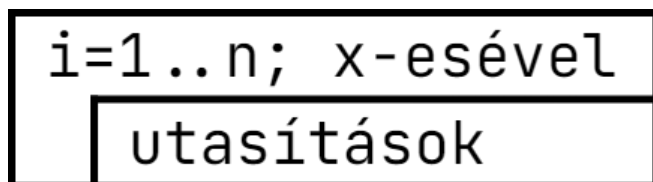


```
do {  
    utasítások  
} while (feltétel);
```

## Számlálós ciklus:



```
for (int i=1;i<=n;++i){  
    utasítások  
}
```



```
for (int i=1;i<=n;i+=x){  
    utasítások  
}
```

# Kód

```
// 1. deklarálás
int n;
int o = 0;
bool van;
// 2. beolvasás
Console.Write("n = ");
int.TryParse(Console.ReadLine(), out n);
// 3. feldolgozás
int i = 2;
while (i < n && n % i != 0) {
    i = i + 1;
}
van = i < n;
if (van) {
    o = i;
}
// 4. kiírás
if (van) {
    Console.WriteLine("o = {0}", o);
}
else {
    Console.WriteLine("Nincs osztó");
}
```

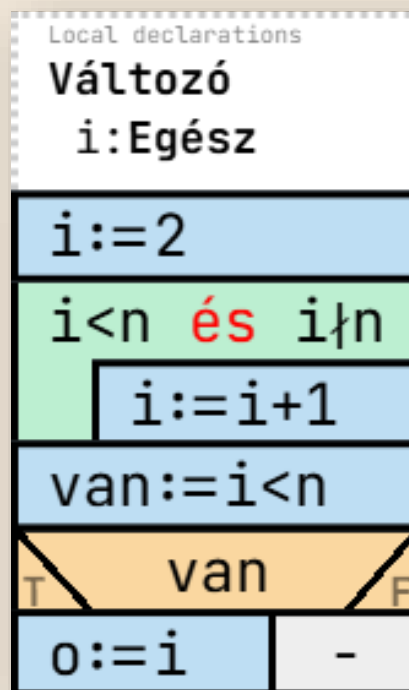
## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$ ,  $van \in \mathbb{L}$

Ef:  $n > 1$

Uf:  $van = \exists i \in [2..n-1] : (i | n) \text{ és } van \rightarrow (2 \leq o < n \text{ és } o | n \text{ és } \forall i \in [2..o-1] : (i \nmid n))$



# Előfeltétel ellenőrzése kódban



# Lehetőségek

Be:  $n$  [ $n > 1$ ]

```
n = 35  
o = 5
```

```
n = 23  
Nincs osztó
```

- Egyszeri beolvasás

- ha jó a beolvasott adat → feldolgozás, kiírás, vége
- ha nem jó → hibaüzenet, vége
- elágazás

```
n = 1  
Nem jó adat!
```

- Ismételt beolvasás

- amíg nem jó a beolvasott adat, addig újra bekérjük
- ciklus

```
n = -1  
Nem jó!  
n = 0  
Nem jó!  
n = 1  
Nem jó!  
n = 2  
Nincs osztó
```

# Egyszeri beolvasás

Be: n [n>1]

```
// 1. deklarálás
int n;
// 2. beolvasás
Console.Write("n = ");
int.TryParse(Console.ReadLine(), out n);
bool jo = n > 1;
if (jo) {
    // 3. feldolgozás
    // ...
    // 4. kiírás
    // ...
}
else {
    Console.WriteLine("Nem jó adat! 1-nél nagyobb egész számot kell megadni!");
}
```

Be: n		Változó jó:Logikai
jó:=n>1		
T	jó	F
Feldolgozás		Ki: "Nem jó"
Kiírás		



# Ismételt beolvasás

// 1. deklarálás

```
int n;
```

// 2. beolvasás

```
bool jo;
```

```
do {
```

```
    Console.WriteLine("n = ");
```

```
    int.TryParse(Console.ReadLine(), out n);
```

```
    jo = (n > 1);
```

```
    if (!jo) {
```

```
        Console.WriteLine("Nem jó!");
```

```
    }
```

```
} while (!jo);
```

// 3. feldolgozás

// ...

// 4. kiírás

// ...

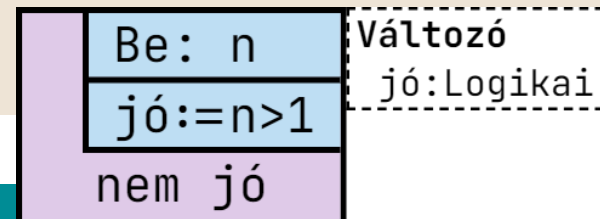
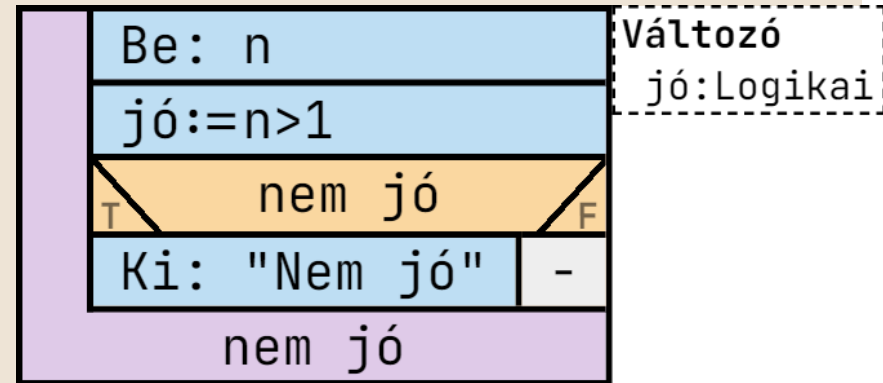
## Specifikáció:

Be:  $n \in \mathbb{N}$

Ki:  $o \in \mathbb{N}$ ,  $van \in \mathbb{L}$

Ef:  $n > 1$

Uf:  $van = \exists i \in [2..n-1] : (i | n) \text{ és } van \rightarrow (2 \leq o < n \text{ és } o | n \text{ és } \forall i \in [2..o-1] : (i \nmid n))$



# Ismételt beolvasás – típushelyesen

```
// 1. deklarálás
int n;
// 2. beolvasás
bool jo;
do {
    Console.Write("n = ");
    jo = int.TryParse(Console.ReadLine(), out n);
    jo = jo && (n > 1);
    if (!jo) {
        Console.WriteLine("Nem jó!");
    }
} while (!jo);
// 3. feldolgozás
// ...
// 4. kiírás
// ...
```

```
n = három
Nem jó!
n = 35ű
Nem jó!
n = 35.3
Nem jó!
n = 35,3
Nem jó!
n = 35
o = 5
```

# Összefoglalás



# Szekvencia

---

## Algoritmus:

utasítás1
utasítás2
utasítás3

## Kód:

```
utasítás1;  
utasítás2;  
utasítás3;
```

# Elágazások

## Algoritmus:

feltétel	
T	F
utasítások1	utasítások2
...	...

kétirányú

\ feltétel1	\ feltétel2	\ ...	\ egyébként
utasítások1	utasítások2	...	utasítások

többsíriányú

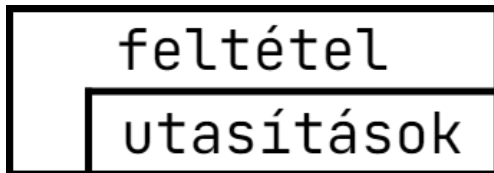
## Kód:

```
if (feltétel) {  
    utasítások1;  
} else {  
    utasítások2;  
}
```

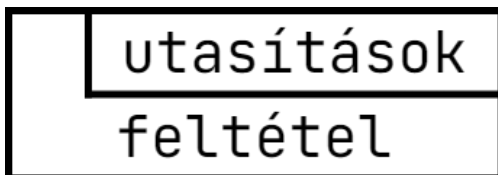
```
if (feltétel1) {  
    utasítások1;  
}  
else if (feltétel2) {  
    utasítások2;  
}  
// ...  
else {  
    utasítások;  
}
```

# Ciklusok

## Feltételes ciklus:

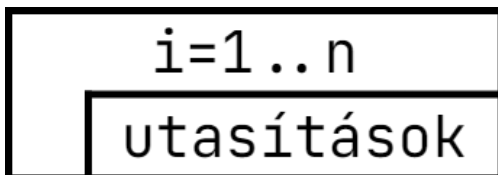


```
while (feltétel){  
    utasítások  
}
```

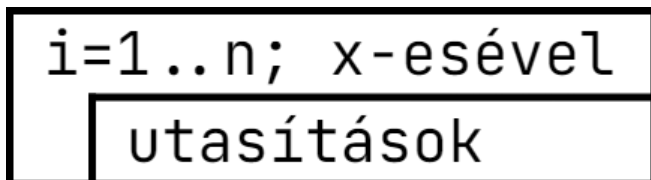


```
do {  
    utasítások  
} while (feltétel);
```

## Számlálós ciklus:



```
for (int i=1;i<=n;++i) {  
    utasítások  
}
```



```
for (int i=1;i<=n;i+=x){  
    utasítások  
}
```

# Ellenőrző kérdések



# Kérdések

---

- Mikor használjunk feltételes és mikor használjunk számlálós ciklust?
- Mi a „filozófiai különbség” a számlálós és a feltételes ciklus között?
- Mi a különbség az előtesztelés és a hátultesztelés ciklus között?
- Hogyan néznek ki pseudokódjai a feltételes ciklusoknak? Add meg az ezeknek megfelelő C# kóddarabokat is!
- Hogyan néznek ki a pseudokódjai a számlálós ciklusnak és az elágazásnak? Add meg az ezek megfelelő C# kóddarabjait is!