

# PROGRAMOZÁS Programozási minták

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 deklarálása, beolvasás, feldolgozás, kiírás



# Feladatmegoldás lépései

- Adat
  - egy-szerű: elemi
  - több különböző: rekord
  - több egyforma: tömb
- Vezérlési szerkezetek-
  - Szekvencia: és
  - Elágazás: ->
  - Ciklus:
- ∀, ∃, ∑

- Feladatmegoldás
  - 1. Példa
  - 2. Specifikáció (← példa)
    - 1. Adatok (Be, Ki)
    - 2. Megszorítás (Ef → Be)
    - 3. Összefüggés (Uf)
  - 3. Adat → Változó
  - ◆4. Algoritmus (← Uf)
    - 5. Kód (← Spec + Alg)

# Megfeleltetések

Példa adat	Specifikáció halmaz	Algoritmus típus	Kód type
3	N	Egész	int
-3	Z	Egész	int
3,3	R	Valós	double
igaz	L	Logikai	bool
"alma"	S	Szöveg	string
"a"	K	Karakter	char
{név:"Győző", jegy: 5}	név:S x jegy:N	Rekord	struct
[3, 5, -6, 2]	Z[1n]	Tömb	int[]

# Analóg programozás



# Analóg problémamegoldás

- Amikor egy olyan feladatot kell megoldani, amelyhez hasonló feladatot már korábban megoldottunk, akkor a megoldó programot a korábbi feladat megoldó programja alapján állítjuk elő.
- Általános problémamegoldó módszer
  - gyakorlati tapasztalat, megtanult ismeret
  - élet minden területén
  - gyakorló programozó eszköztárában is



# Analóg problémamegoldás – példák

- Szerelés
- Levendulaültetés
- Daruzás
- Pelenkacsere

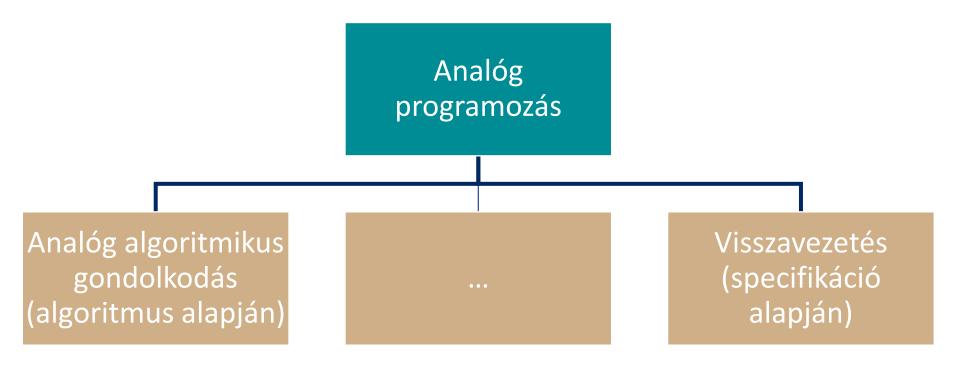




## Analóg programozás

- Cél: egy feladat megoldására program készítése
- Minden egyes alkalommal újra és újra kitalálhatjuk a megoldást
- Vagy egy korábbi, hasonló feladat megoldását vehetjük alapul
- Analóg programozás: a konkrét feladatot egy korábbi feladat megoldása alapján állítjuk elő.

# Analóg programozás



# Analóg programozás – analóg algoritmikus gondolkodás

Konkrét feladat Mintafeladat (feladatosztály sablonja) Specifikáció Specifikáció Algoritmus **Algoritmus** Kód A mintafeladathoz hasonló feladatokat a

A mintafeladathoz hasonlo feladatokat a mintafeladatnál alkalmazott gondolatsorral, annak ötleteit kölcsönözve, lemásolva, analóg algoritmikus gondolkodással oldjuk meg.

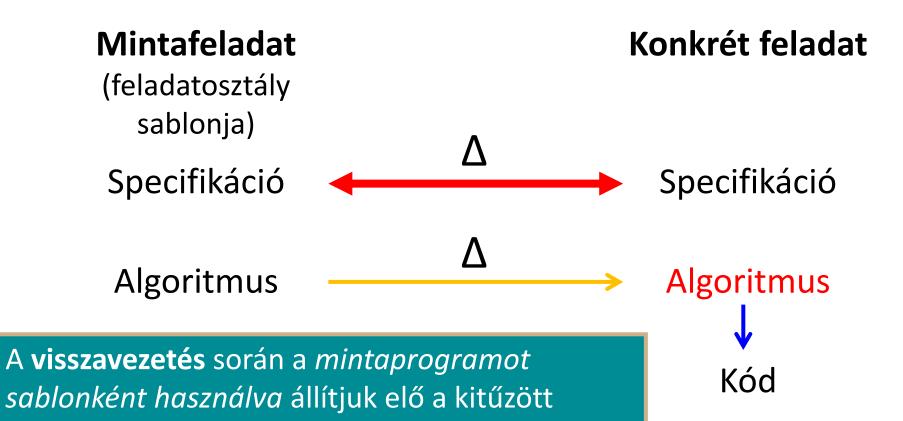


# Analóg programozás – visszavezetés

feladat megoldó programját úgy, hogy

specifikációkban felfedett eltéréseket.

figyelembe vesszük a kitűzött- és a mintafeladat





### Visszavezetés

- A visszavezetés során a mintafeladat megoldó programját (a mintaprogramot) sablonként használva állítjuk elő a kitűzött feladat megoldó programját úgy, hogy figyelembe vesszük a kitűzött- és a mintafeladat specifikációkban felfedett eltéréseket.
- Megtehetjük, mivel ismerjük a mintafeladat és a kitűzött feladat precíz leírását.

## Példa – "mintafeladat"

#### **Feladat:**

Egy középiskolai osztályban papírgyűjtést szerveznek. Minden diákról tudjuk, hány kg papírt hozott be. Határozd meg, hogy mennyi gyűlt összesen össze?

Példa: [1, 3, 2, 0, 5, 2] → 13

Példa: n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

-	kg
1	1
1 2 3	3
3	2
4	0
5	5
n=6	2
_	
össz=	13
•	

## Példa – "mintafeladat"

#### **Feladat:**

Példa: n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

Határozd meg egy n elemű tömb elemeinek összegét!

## **Specifikáció:**

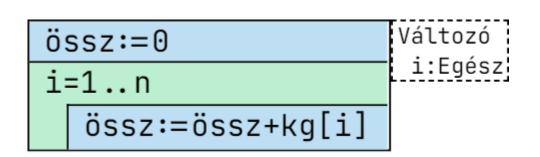
Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

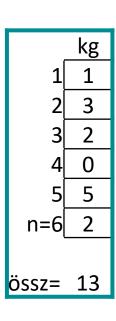
Ki: össz∈N

Ef: -

Uf: össz = SZUMMA(i=1..n, kg[i])

### **Algoritmus:**





# Példa – "mintafeladat" algoritmikus gondolkodással

#### **Feladat:**

Példa: n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

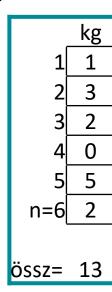
Határozd meg egy n elemű tömb elemeinek összegét!

### **Algoritmus:**



### Leírás:

A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a tömb indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított össz változóhoz a kg[i] értéket. A megoldó program tehát a kezdeti értékadás után (ahol az össz változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

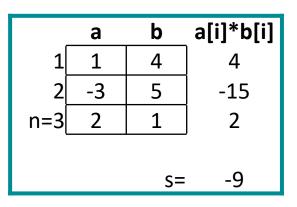


### Példa – skaláris szorzat

#### **Feladat:**

Határozzuk meg két azonos dimenziójú vektor skaláris szorzatát!

```
Példa:
n=3,
a=[1, -3, 2], → s = -9
b=[4, 5, 1]
```



### Példa – skaláris szorzat

Példa: n=3,  $a=[1, -3, 2], \rightarrow s=-9$ b=[4, 5, 1]

#### **Feladat:**

Határozzuk meg két azonos dimenziójú vektor skaláris

szorzatát!

## **Specifikáció:**

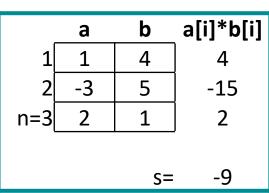
Be:  $n \in \mathbb{N}$ ,  $a \in \mathbb{Z}[1..n]$ ,  $b \in \mathbb{Z}[1..n]$ 

Ki: s∈Z

Ef: -

Uf: s = SZUMMA(i=1...n, a[i]\*b[i])

$$s = \sum_{i=1}^{n} a[i] * b[i]$$



#### Példa – skaláris szorzat

## Papírgyűjtés

Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n,kg[i])

### Skaláris szorzat

Be:  $n \in \mathbb{N}$ ,  $a \in \mathbb{Z}[1...n]$ ,  $b \in \mathbb{Z}[1...n]$ 

Ki: s∈Z

Ef: -

Uf: s=SZUMMA(i=1..n,a[i]\*b[i])

A specifikációk összevetéséből is látszik, hogy a két feladat hasonlít egymásra.

Állítsuk elő az algoritmust

- analóg algoritmikus gondolkodással, majd
- visszavezetéssel!

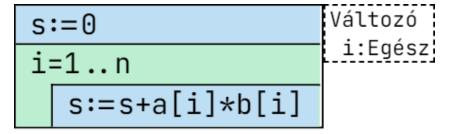


## Példa – skaláris szorzat analóg algoritmikus gondolkodás

	а	b	a[i]*b[i]
1	1	4	4
2	-3	5	-15
n=3	2	1	2
Ì			
		s=	-9

### Algoritmus:





### Leírás:



A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a tömb indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított össz változóhoz a kg[i] értéket. A megoldó program tehát a kezdeti értékadás után (ahol az össz változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.



A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót a tömbök indexein, és minden lépésben hozzá kell adni a kezdetben nullára beállított s változóhoz az a[i]\*b[i] értéket. A megoldó program tehát a kezdeti értékadás után (ahol az s változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.

# Példa – skaláris szorzat visszavezetéssel Vegyük számba

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

# Papírgyűjtés (mintafeladat)

Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n,kg[i])

# Skaláris szorzat (konkrét feladat)

Be:  $n \in \mathbb{N}$ ,  $a \in \mathbb{Z}[1..n]$ ,  $b \in \mathbb{Z}[1..n]$ 

Ki: s∈Z

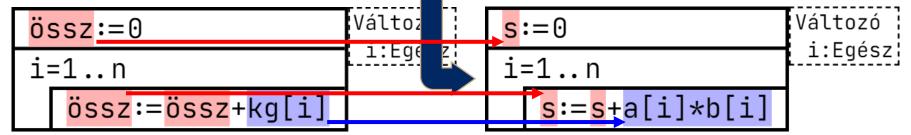
Ef: -

Uf: s=SZUMMA(i=1..n,a[i]\*b[i])

össz ~ s

 $kg[i] \sim a[i]*b[i]$ 

## **Algoritmus:**



## Példa – számok összege

#### **Feladat:**

Határozzuk meg az a és b egész számok (a≤b) közé eső egész számok összegét (a határokat is beleértve)!

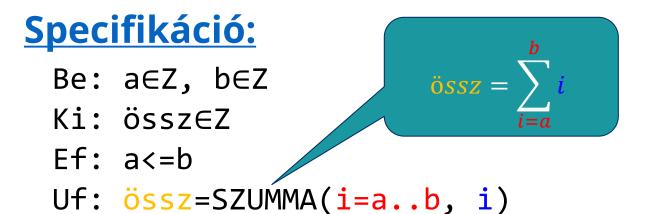
```
Példa:
a=1, b=5 → össz=15
a=-2, b=2 → össz=0
```

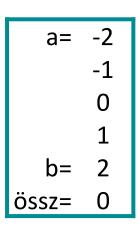
## Példa – számok összege

Példa: a=1, b=5 → össz=15 a=-2, b=2 → össz=0

#### **Feladat:**

Határozzuk meg az a és b egész számok (a≤b) közé eső egész számok összegét (a határokat is beleértve)!





## Példa – számok összege

## Papírgyűjtés

(mintafeladat)

Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n,kg[i])

## Számok összege

(konkrét feladat)

Be:  $a \in Z$ ,  $b \in Z$ 

Ki: össz∈Z

Ef: a<=b

Uf: össz=SZUMMA(i=a..b, i)

A specifikációk összevetéséből is látszik, hogy a két feladat hasonlít egymásra.

Állítsuk elő az algoritmust visszavezetéssel!

# Példa – számok összege visszavezetéssel Vegyük számba a

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

## Papírgyűjtés

(mintafeladat)

Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n,kg[i])

## Számok összege

(konkrét feladat)

Be:  $a \in Z$ ,  $b \in Z$ 

Ki: össz∈Z

Ef: a<=b

Uf: össz=SZUMMA(i=a..b, i)

1..n ~ a..b kg[i] ~ i

#### **Algoritmus:**

```
      össz:=0
      Válto i:Eg

      i=1..n
      össz:=össz+kg[i]
```

```
össz:=0

i=a..b

össz:=össz+i
```

# Tanulságok

- Eltérések ellenére ezek a feladatok annyira hasonlóak, hogy ugyanazon sablon alapján megoldhatók.
- Mi alapján sorolhatók ugyanazon sablon alá?
- Hogyan lehetne általánosan megfogalmazni az ilyen feladatokat?

## Közös tulajdonságok egész számok zárt intervalluma

#### **Papírgyűjtés**

#### Skalárszorzat

#### Számok összege

	kg		i		kg[i]
1	1	e=	1	$\rightarrow$	1
2	3		2	$\rightarrow$	3
3	2		3	$\rightarrow$	2
4	0		4	$\rightarrow$	0
5	5		5	$\rightarrow$	5
6	2	u=	6	$\rightarrow$	2

	а	b		i		a[i]*b[i]
1	1	4	e= :	1	$\rightarrow$	4
2	-3	5		2	$\rightarrow$	-15
3	2	1	u= 3	3	$\rightarrow$	2

 $\begin{array}{cccc} & i & & i \\ e = -2 & \to & 1 \\ & -1 & \to & 3 \\ & 0 & \to & 2 \\ & 1 & \to & 0 \\ u = 2 & \to & 5 \end{array}$ 

Ciklusváltozó futási tartományát határozza meg

Tömb indextartománya

Tömb indextartománya

Zárt intervallum



### Közös tulajdonságok intervallum elemeihez rendelt érték

#### **Papírgyűjtés**

#### Skalárszorzat

#### Számok összege

össz=SZUMMA(i=a..b, i)

	kg		i		kg[i]
1	1	e=	1	$\rightarrow$	1
2	3		2	$\rightarrow$	3
3	2		3	$\rightarrow$	2
4	0		4	$\rightarrow$	0
5	5		5	$\rightarrow$	5
6	2	u=	6	$\rightarrow$	2

	а	b		i		a[i]*b[i]
1	1	4	e=	1	$\rightarrow$	4
2	-3	5		2	$\rightarrow$	-15
3	2	1	u=	3	$\rightarrow$	2

Tömb eleme

Tömbök elemértékének szorzata

Az intervallum i eleme



## Közös tulajdonságok összegzés

#### **Papírgyűjtés**

#### Skalárszorzat

#### Számok összege

```
össz=SZUMMA(i=1..n, kg[i])
s=SZUMMA(i=1..n, a[i]*b[i])
```

ossz=SZUMMA(i=a..b, i)

,	kg	i kg[i]
1	1	$e=1 \rightarrow 1$
2	3	$2 \rightarrow \frac{3}{3}$
3	2	$3 \rightarrow \frac{2}{2}$
4	0	$4 \rightarrow 0$
5	5	$5 \rightarrow \frac{5}{5}$
6	2	$u= 6 \rightarrow \frac{1}{2}$

A leképezett értékeket 0-tól kezdve össze kellett adni: s=0+f(e)+f(e+1)+...+f(u)

i		i
e= -2	$\rightarrow$	1
-1	$\rightarrow$	1 3
0	$\rightarrow$	2
1	$\rightarrow$	Ŏ
u= 2	$\rightarrow$	5

ÖSSZ S ÖSSZ

## Általános feladat

#### **Feladat:**

Legyen adott az egész számok egy [e..u] intervallumán értelmezett f:[e..u]→H függvény (H olyan halmaz, amelyen értelmezett az összeadás művelete). Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz az f(e)+f(e+1)+...+f(u) kifejezés értékét!

## Általános feladat

#### Feladat:

Legyen adott az egész számok egy [e..u] intervallumán értelmezett f:[e..u]→H függvény (H olyan halmaz, amelyen értelmezett az összeadás művelete). Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz az f(e)+f(e+1)+...+f(u) kifejezés értékét!

### Specifikáció:

Be: e∈Z, u∈Z

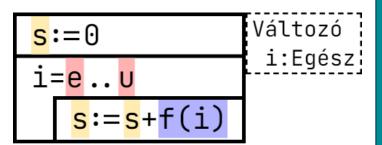
Ki: s∈H

Ef: -

Uf: s=SZUMMA(i=e..u, f(i))

i		f(i)
е	$\rightarrow$	1
e+1	$\rightarrow$	3
•••	$\rightarrow$	3 2
u-1	$\rightarrow$	Ō
u	$\rightarrow$	5

## **Algoritmus:**



A megoldás során a kívánt összeget fokozatosan állítjuk elő. Ehhez végig kell vezetnünk egy i egész típusú változót az intervallum elemein, és minden lépésben hozzá kell adni a kezdetben nullára beállított s változóhoz az f(i) értéket. A megoldó program tehát a kezdeti értékadás után (ahol az s változó értékét nullára állítjuk) egy ciklus, amely az i változót egyesével növeli egészen n-ig.



## Példa – papírgyűjtés

#### **Feladat:**

Példa: n=6, kg=[1, 3, 2, 0, 5, 2] → össz=13

Határozd meg egy n elemű tömb elemeinek összegét!

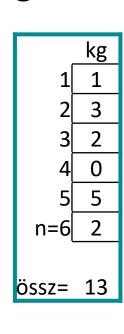
## Specifikáció:

```
Be: n \in \mathbb{N}, kg \in \mathbb{N}[1..n]
```

Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n, kg[i])



# Példa – papírgyűjtés visszavezetés Vegyüks

Vegyük számba a két feladat közötti különbségeket, a mintaprogram algoritmusában pedig cseréljük le a megfelelő részeket!

#### **Feladatsablon**

(mintafeladat)

Be: e∈Z, u∈Z

Ki: s∈H

Ef: -

Uf: s=SZUMMA(i=e..u, f(i))

### Papírgyűjtés

(konkrét feladat)

Be:  $n \in \mathbb{N}$ ,  $kg \in \mathbb{N}[1..n]$ 

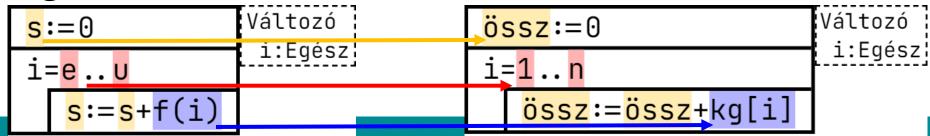
Ki: össz∈N

Ef: -

Uf: össz=SZUMMA(i=1..n, kg[i])

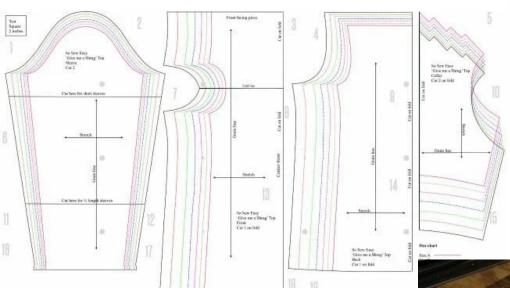
#### Visszavezetés:

#### **Algoritmus:**



# Nevezetes programozási minták Programozási tételek

## Hétköznapi példa







## Analóg programozás

- Cél: egy feladat megoldására program készítése
- Minden egyes alkalommal újra és újra kitalálhatjuk a megoldást
- Vagy egy korábbi, hasonló feladat megoldását vehetjük alapul
- Analóg programozás: a konkrét feladatot egy korábbi feladat megoldása alapján állítjuk elő visszavezetéssel.

# Feladattípusok

- A programok ad hoc módon is előállhatnak.
- De: felismerhetjük, hogy vannak megoldások, amelyek többé-kevésbé ugyanazt a feladatot oldják meg csak más és más köntösben
- Feladattípusok
- Ezekhez készíthetünk/létezik bizonyítottan helyes megoldás
- Analóg programozás során ezeket használjuk mintának.

# Programozási minta Programozási tétel

# Célja:

Bizonyíthatóan **helyes sablon**, amelyre magasabb szinten lehet építeni a megoldást. (A fejlesztés gyorsabb és biztonságosabb.)

### Szerkezete:

- 1.absztrakt feladat specifikáció
- 2.absztrakt algoritmus

# Egy fontos előzetes **megjegyzés**:

A bemenet legalább egy sorozat...

# Programozási minta

### Felhasználásának menete:

- 1.a konkrét feladat specifikálása
- 2.a specifikációban a programozási minták (PrM-ek) megsejtése
- 3.a konkrét feladat és az absztrakt feladat paramétereinek egymáshoz rendelése
- 4.a konkrét algoritmus "generálása" a megsejtett PrM-ek absztrakt algoritmusok alapján, 3. szerint átparaméterezve
- 5. "hatékonyítás" programtranszformációkkal

# Összegzés



# Összegzés

### **Feladatok:**

- Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy év végére mennyivel nőtt a vagyona!
- 2. Ismerjük egy autóversenyző körönkénti idejét. Adjuk meg az **átlag**körének idejét!
- 3. Adjuk meg az N számhoz az N faktoriális értékét!
- 4. Ismerjük egy iskola szakkö Mi bennük a közös? szakkörönként. Adjuk meg n szám összegét kell kiszámolni!
- 5. Ismerünk N szót. Adjuk meg a belőlük összeállított mondatot!

# Összegzés sablon

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u] $\rightarrow$ H függvény. A H halmaz elemein értelmezett az összeadás művelet. Határozzuk meg az f függvény [e..u] intervallumon felvett értékeinek az összegét, azaz a  $\sum_{i=e}^{u} f(i)$  kifejezés értékét! (e>u esetén ennek az értéke definíció szerint a nulla elem)

# Specifikáció

Be: e∈Z, u∈Z

Ki: s∈H

Ef: -

Uf: s=SZUMMA(i=e..u, f(i))

```
s:=0
    i=e..u
    s:=s+f(i)
Változó
    i:Egész
```

# Példa – jövedelmek visszavezetés

Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy év végére **mennyivel** nőtt a vagyona!

### **Feladatsablon**

(mintafeladat)

Be: e∈Z, u∈Z

Ki: s∈H

Ef: -

Uf: s=SZUMMA(i=e..u, f(i))

### <u>Jövedelmek</u>

(konkrét feladat)

Be: n∈N, jöv∈Jövedelem[1..n],

Jövedelem=(be:N x ki:N)

Ki: s∈Z

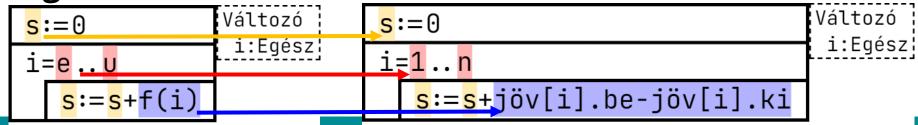
Ef: -

Uf: s=SZUMMA(i=1..n,

jöv[i].be-jöv[i].ki)

### Visszavezetés:

e..u ~ 1..n
f(i) ~ jöv[i].be-jöv[i].ki



# Megszámolás



# Megszámolás

### Mi bennük a közös?

n darab "valamire" kell megadni, hogy hány adott tulajdonságú van közöttük!

### **Feladatok:**

- 1. Ismerjük egy ember havi bevételeit és kiadásait. Adjunk meg, hogy **hány** hónapban nőtt a vagyona!
- 2. Adjuk meg egy természetes szám osztói **számá**t!
- 3. Adjuk meg egy ember nevében levő "a" betűk **számá**t!
- 4. Adjunk meg az éves statisztika alapján, hogy **hány** napon fagyott!
- 5. Adjuk meg n születési hónap alapján, hogy közöttük **hány**an születtek télen!

# Példa – a betűk száma algoritmikus gondolkodással

# i név név[i]="a" érték 1 a IGAZ 1 2 I HAMIS 0 3 m HAMIS 0 4 a IGAZ 1 III db= 2

### **Feladat:**

Adjuk meg egy ember nevében levő "a" betűk **számá**t!

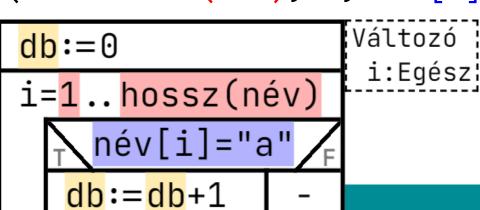
# **Specifikáció:**

Be: név∈S

Ki: db∈N

Ef: -

Uf: db=SZUMMA(i=1..hossz(név), 1, név[i]="a")



# Megszámolás sablon

## i T(i) érték e IGAZ 1 e+1 HAMIS 0 ... HAMIS 0 u IGAZ 1 u db= 2

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy az [e..u] intervallumon a T feltétel hányszor veszi fel az igaz értéket!

### Specifikáció

Be: e∈Z, u∈Z

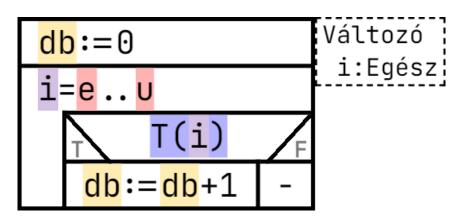
Ki: db∈N

Ef: -

Uf: db=SZUMMA(i=e..u, 1, T(i))

Rövidítve:

Uf: db=DARAB(i=e..u, T(i))



# Példa – téli születések visszavezetés

Adjuk meg n születési hónap alapján, hogy közöttük hányan születtek télen!

### **Feladatsablon**

(mintafeladat)

Be: e∈Z, u∈Z

Ki: db∈N

Ef: -

Uf: db=DARAB(i=e..u, T(i))

### Téli születések száma

(konkrét feladat)

Be:  $n \in \mathbb{N}$ ,  $h \circ \in \mathbb{N}[1..n]$ 

Ki: db∈N

Ef:  $\forall i \in [1..n]: (1 < = ho[i] < = 12)$ 

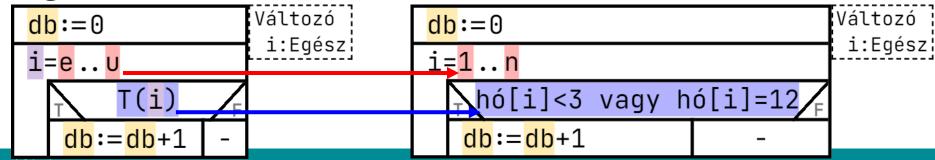
Uf: db=DARAB(i=1..n,

hó[i]<3 vagy hó[i]=12)

### Visszavezetés:

```
e..u ~ 1..n
T(i) ~ hó[i]<3 vagy hó[i]=12</pre>
```

Megjegyzés: a konkrét feladat előfeltétele mindig lehet szigorúbb a tétel előfeltételéné!!



# Maximumkiválasztás



### Maximumkiválasztás

### Feladatok:

- 1. Ismerjük egy ember havi bevételeit és kiadásait. Adjunk meg, hogy melyik hónapban nőtt **leg**jobban a vagyona!
- 2. Adjuk meg n ember közül az ábécében **utolsó**t!
- 3. Adjuk meg n ember közül azt, aki a legtöbb ételt szereti!
- 4. Adjuk meg n napi statisztika alapján a **leg**melegebb napot!
- Adjuk meg n születésnap alapján azt, akinek idén először van születésnapja!

### Mi bennük a közös?

n darab "valami" közül kell megadni a legnagyobbat (vagy a legkisebbet)!

#### Fontos:

A "valamik" között értelmezhető egy **rendezési reláció**. Ha **legalább 1** "valamink" van, akkor legnagyobb (legkisebb) is biztosan van közöttük!



# Példa – legmelegebb nap algoritmikus gondolkodással

	nap	nap	nap	nap
i hőm	(i=1)	(i=2)	(i=3)	(i=4)
1 13,5	13,5	13,5	13,5	13,5
2 12,6	12,6	12,6	12,6	12,6
3 14,8	14,8	14,8	14,8	14,8
4 10,2	10,2	10,2	10,2	10,2

### **Feladat:**

Adjuk meg n napi statisztika alapján a legmelegebb napot!

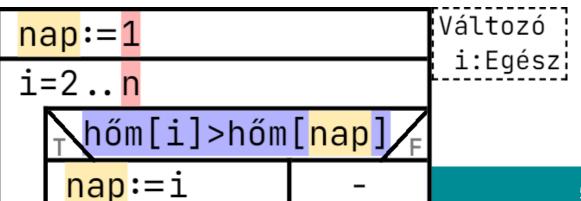
# **Specifikáció:**

```
Be: n∈N, hőm∈R[1..n]
```

Ki: nap∈N

Ef: n>0 és  $\forall i \in [1..n]: (-100 < = hőm[i] < = 100)$ 

Uf:  $nap \in [1..n]$  és  $\forall i \in [1..n]: (hőm[nap] > = hőm[i])$ 





# Maximumkiválasztás sablon

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az f függvény hol veszi fel az [e..u] nem üres intervallumon a legnagyobb értéket, és mondjuk meg, mekkora ez a maximális érték!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: maxind∈Z, maxért∈H
Ef: e<=u
Uf: maxind∈[e..u] és
∀i∈[e..u]:(f(maxind)>=f(i)) é
maxért=f(maxind)
```

### **Algoritmus**

#### Rövidítve:

```
Uf: (maxind, maxért) = MAX(i=e..u, f(i))
```

# Példa – ábécében utolsó visszavezetés

Adjuk meg n ember közül az ábécében utolsót!

### **Feladatsablon**

Be: e∈Z, u∈Z

Ki: maxind∈Z, maxért∈H

Ef: e<=u

Uf: (maxind, maxért)=

MAX(i=e..u,f(i))

## Ábécében utolsó név

Be: n∈N, nevek∈S[1...n]

Ki: utind∈N, utnév∈S

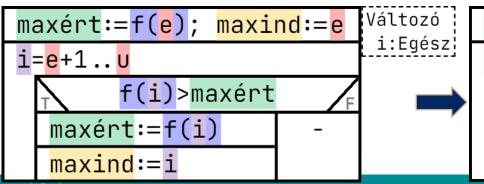
Ef: n>0

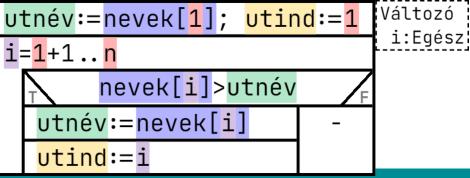
Uf: (utind,utnév)=

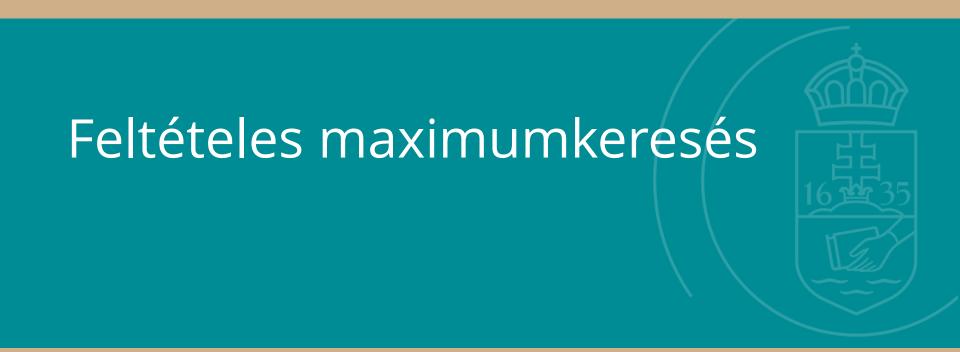
MAX(i=1..n,nevek[i])

### Visszavezetés:

```
maxind, maxért ~ utind, utnév
e..u ~ 1..n
f(i) ~ nevek[i]
```







### Feltételes maximumkeresés

### **Feladatok:**

- Ismerjük egy ember havi bevételeit és kiadásait. Adjuk meg, hogy melyik hónapban volt a legnagyobb vesztesége!
- 2. Adjuk meg n napi statisztika alapján a **leg**melegebb **fagyos** napot!
- 3. Egy hibakezelő szoftverben számokkal jelezzük a hibák súlyosságát. Adjuk meg a **leg**súlyosabb hibát (**ha van**)!

### Mi bennük a közös?

n darab "valami" adott tulajdonságú elemei közül kell megadni a legnagyobbat (vagy a legkisebbet)!

#### Fontos:

A "valamik" között értelmezhető egy rendezési reláció.

Nem biztos, hogy van adott tulajdonságú "valami".

Nem biztos, hogy egyáltalán van "valami"-nk.



# Példa – legmelegebb fagyos nap algoritmikus gondolkodással

Feladat: Adjuk meg n napi statisztika alapján a legmelegebb fagyos napot!

```
i hóm van? maxért?
1 2,3 HAMIS ???
2 -1,3 IGAZ -1,3
3 -0,5 IGAZ -0,5
4 0,4 IGAZ -0,5
```

```
i hốm van? maxért?
1 2,3 HAMIS ???
2 1,3 HAMIS ???
3 0,5 HAMIS ???
4 0,4 HAMIS ???
```

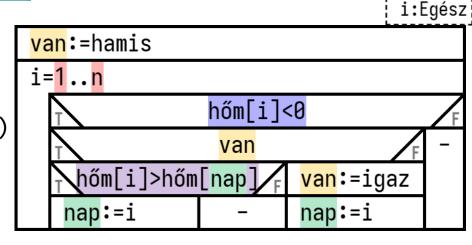
### Specifikáció és algoritmus:

```
Be: n∈N, hốm∈R[1..n]
Ki: van∈L, nap∈N
Ef: ∀i∈[1..n]:(-100<=hốm[i]<=100)</pre>
```

Uf:  $van=\exists i \in [1..n]: (hőm[i]<0)$  és

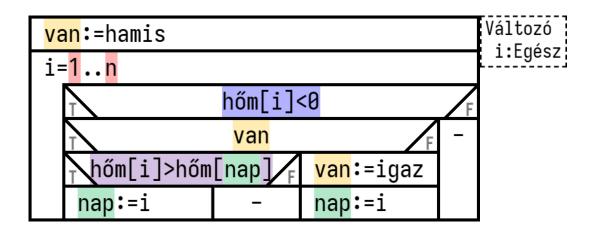
```
van->(nap∈[1..n] és
hőm[nap]<0 és
```

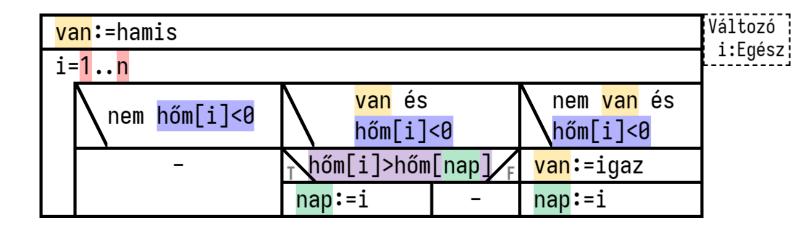
```
\forall i \in [1..n]: (hőm[i] < 0 \rightarrow hőm[nap] > = hőm[i]))
```



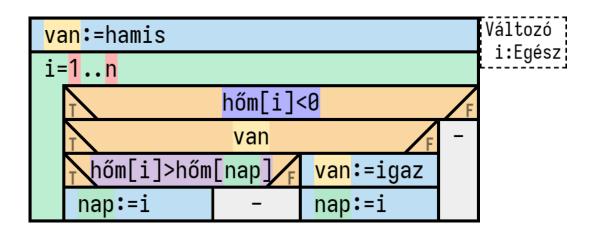
Változó

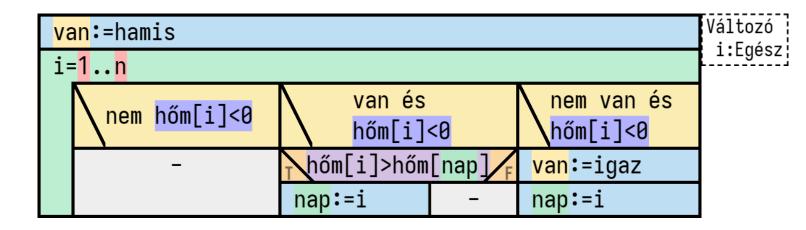
# Példa – legmelegebb fagyos nap algoritmikus gondolkodással – átalakítás





# Példa – legmelegebb fagyos nap algoritmikus gondolkodással – átalakítás





# Feltételes maximumkeresés sablon

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma, egy f:[e..u]→H függvény és egy T:[e..u]→Logikai feltétel. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az [e..u] intervallum T feltételt kielégítő elemei közül az f függvény hol veszi fel a legnagyobb értéket, és mondjuk meg,

mekkora ez az érték!

### Specifikáció és algoritmus:

```
Be: e∈Z, u∈Z
```

Ki: van∈L, maxind∈Z, maxért∈H

Ef: -

Uf:  $van = \exists i \in [e..u]:(T(i))$  és  $van \rightarrow (maxind \in [e..u])$  és

```
maxért=f(maxind) és T(maxind) és
∀i∈[e..u]:(T(i) -> maxért>=f(i)))
```

#### Rövidítve:

Uf: (van, maxind, maxért) = FELTMAX(i = e..u, f(i), T(i))



i:Egész

# Feltételes maximumkeresés sablon

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma, egy f:[e..u]→H függvény és egy T:[e..u]→Logikai feltétel. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg, hogy az [e..u] intervallum T feltételt kielégítő elemei közül az f függvény hol veszi fel a legnagyobb értéket, és mondjuk meg, Változó mekkora ez az érték!

### Specifikáció és algoritmus:

```
Be: e∈Z, u∈Z
```

Ki: van∈L, maxind∈Z, maxért∈H

Ff: -

Uf:  $van = \exists i \in [e..u]:(T(i))$  és van -> (maxind∈[e..u] és

van:=hamis i=e..u van és nem van és nem T(i) T(i)T(i)f(i)>maxért van:=igaz maxért:=f(i) maxért:=f(i) maxind:=i maxind:=i

```
maxért=f(maxind) és T(maxind) és
\forall i \in [e..u]:(T(i) \rightarrow maxért \rightarrow f(i)))
```

#### Rövidítve:

Uf: (van, maxind, maxért) = FELTMAX(i = e..u, f(i), T(i))



i:Egész

# Példa – súlyos hiba visszavezetés

Egy hibakezelő szoftverben számokkal jelezzük a hibák súlyosságát. Adjuk meg a legsúlyosabb hibát (ha van)!

### Feladatsablon

# Legsúlyosabb hiba

Be: n∈N, hibák∈Hiba[1..n], Hiba=(név:S x súly:N)

Sa: maxind∈N, maxért∈N

Ki: van∈L, lsh∈S

Ef:  $\forall i \in [1..n]: (1 < = hibák[i].súly < = 5)$ 

van

Uf: (van, maxind, maxért)=

FELTMAX(i=e..u,f(i),T(i)) FELTMAX(i=1..n,hibák[i].súly,igaz) és

### Visszavezetés: e..u ~ 1..n

**Algoritmus:** 

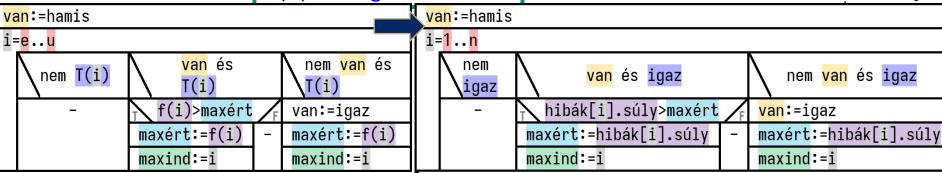
f(i) ~ hibák[i].súly

i(i) ~ nibak[i].Sul

T(i) ~ igaz

van->lsh=hibák[maxind].név

Változó i:Egész, maxind:Egész, maxért:Egész



lsh:=hibák[maxind].név



# Példa – súlyos hiba visszavezetés

Egy hibakezelő szoftverben számokkal jelezzük a hibák súlyosságát. Adjuk meg a legsúlyosabb hibát (ha van)!

### **Feladatsablon**

```
Be: e \in Z, u \in Z
Ki: van∈L, maxind∈Z, maxért∈H
Ef: -
Uf: (van, maxind, maxért)=
             FELTMAX(i=e..u,f(i),T(i))
```

# Legsúlyosabb hiba

Be: n∈N, hibák∈Hiba[1..n], Hiba=(név:S x súly:N)

Sa: maxind∈N

lsh:=hibák[maxind].név

Ki: van∈L, lsh∈S

Ef:  $\forall i \in [1..n]: (1 < = hibák[i].súly < = 5)$ 

Uf: (van, maxind,)=

FELTMAX(i=1..n, hibák[i].súly, igaz) és

### Visszavezetés: e..u ~ 1..n

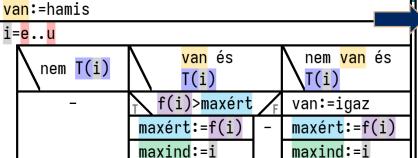
**Algoritmus:** 

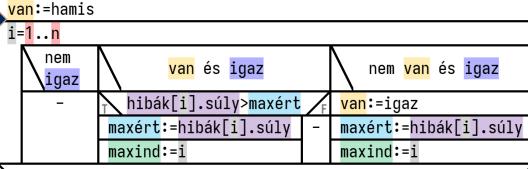
f(i) ~ hibák[i].súly

 $T(i) \sim igaz$ 

van->lsh=hibák[maxind].név

Változó i:Eqész, maxind:Egész, maxért:Egész





van



# Keresés



### Keresés

### Mi bennük a közös?

N darab "valami" közül kell megadni egy adott tulajdonságút, ha nem tudjuk, hogy ilyen elem van-e!

### **Feladatok:**

- Ismerjük egy ember havi bevételeit és kiadásait. Év végére nőtt a vagyona. Adjunk meg egy hónapot, amikor nem nőtt a vagyona!
- 2. Adjuk meg egy természetes szám egy 1-től és önmagától különböző legkisebb osztóját!
- 3. Adjuk meg egy ember nevében egy "a" betű helyét!
- **4. Adjunk meg egy** tanulóra egy tárgyat, amiből megbukott!
- **5. Adjuk meg egy** számsorozat olyan elemét, amely nagyobb az előzőnél!

# Példa – legkisebb osztó algoritmikus gondolkodással

### **Feladat:**

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

### Specifikáció és algoritmus:

Be: n∈N

Ki: o∈N, van∈L

Ef: n>1

Uf:  $van=\exists i \in [2..n-1]:(i|n)$  és van->(2<=o< n és o|n és

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

```
i i<=8? i∤n?
1
e= 2 IGAZ IGAZ
3 IGAZ HAMIS
4
5
6
7
u= 8
n= 9</pre>
```

```
i i<=4? i∤n?

1
e= 2 IGAZ IGAZ
3 IGAZ IGAZ
u= 4 IGAZ IGAZ
n= 5 HAMIS
```

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: van=∃i∈[e..u]:(T(i)) és
    van->(ind∈[e..u] és T(ind) és
    ∀i∈[e..ind-1]:(nem T(i)))
```

#### Rövidítve:

```
Uf: (van,ind)=KERES(i=e..u,T(i))
```

```
i:=e
i ≤ u és nem T(i)
i:=i+1
van:=i ≤ u

r van
ind:=i -
```

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

### Specifikáció

Uf: (van,ind)=KERES(i=e..u,T(i))

```
ind:=e
ind ≤ u és nem T(ind)
ind:=ind+1
van:=ind ≤ u
```

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

### Specifikáció

```
van:=hamis; ind:=e
nem van és ind ≤ u

T(ind)
van:=igaz ind:=ind+1
```

### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg az [e..u] intervallumban balról az első olyan számot, ha van, amely kielégíti a T feltételt!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: van∈L, ind∈Z
Ef: -
Uf: van=∃i∈[e..u]:(T(i)) és
     van->(ind∈[e..u] és T(ind) és
     ∀i∈[e..ind-1]:(nem T(i)))
```

### **Algoritmus**

```
van:=hamis; i:=e
nem van és i ≤ u
van:=T(i)
ind:=i
i:=i+1
```

### Rövidítve:

```
Uf: (van,ind)=KERES(i=e..u,T(i))
```

# Példa – legkisebb osztó visszavezetés

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

### **Feladatsablon**

#### Be: e∈Z, u∈Z

Ki: van∈L, ind∈Z

Ef: -

### Legkisebb osztó

Be: n∈N

Ki: o∈N, van∈L

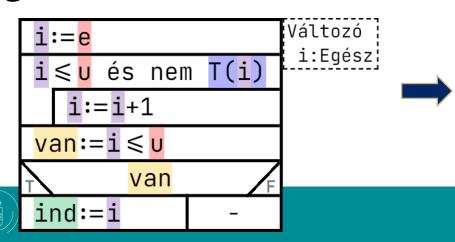
Ef: n>1

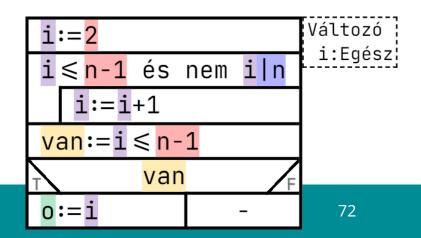
Uf: (van, ind) = KERES(i=e..u, T(i)) Uf: (van, o) = KERES(i=2..n-1, i|n)

### Visszavezetés:

```
ind
e..u ~ 2..n-1
         i|n
T(i)
```

Megjegyzés: a konkrét feladat előfeltétele mindig lehet szigorúbb a tétel előfeltételénél!





# Példa – legkisebb osztó visszavezetés

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

### Feladatsablon

### Be: e∈Z, u∈Z

Ki: van∈L, ind∈Z

Ef: -

CI. -

### Legkisebb osztó

Be: n∈N

Ki: o∈N, van∈L

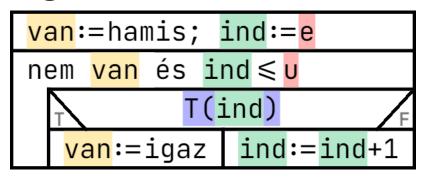
Ef: n>1

Uf: (van, ind) = KERES(i=e..u, T(i)) Uf: (van, o) = KERES(i=2..n-1, i|n)

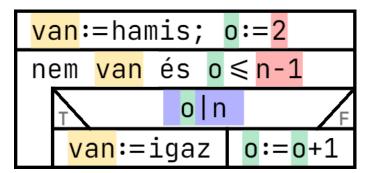
### Visszavezetés:

```
ind \sim 0
e..u \sim 2..n-1
T(i) \sim i|n
```

Megjegyzés: a konkrét feladat előfeltétele mindig lehet szigorúbb a tétel előfeltételénél!







# Eldöntés



### Eldöntés

#### Mi bennük a közös?

Döntsük el, hogy N "valami" között van-e adott tulajdonsággal rendelkező elem!

Ez a keresés programozási tétel (kimenetének) szűkítése.

### **Feladatok:**

- 1. Egy természetes számról **döntsük el**, hogy prímszám**-e**!
- 2. Egy szóról **mondjuk meg**, hogy egy hónapnak a neve-**e**!
- 3. Egy tanuló év végi osztályzatai alapján **állapítsuk meg**, hogy bukott-**e**!
- 4. Egy szóról adjuk meg, hogy van-e benne magánhangzó!
- 5. Egy számsorozatról **döntsük el**, hogy monoton növekvő**- e**!
- Egy tanuló év végi jegyei alapján adjuk meg, hogy kitűnőe!

### Példa – hónapnév algoritmikus gondolkodáss

#### **Feladat:**

Egy szóról **mondjuk meg**, hogy egy hónapnak a neve**-e**!

### **Specifikáció:**

Be: név∈S, hónév∈S[1..12]=["januar","+ebruar",...

Ki: hónapnéve∈L

Ef: -

Uf: hónapnéve=∃i∈[1..12]:(hónév[i]=név)

### **Algoritmus:**

```
i:=1
i ≤ 12 és nem hónév[i]=név
i:=i+1
hónapnéve:=i ≤ 12
Változó
i:Egész
i
```





### Eldöntés sablon

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy van-e az [e..u] intervallumnak olyan eleme, amely kielégíti a T feltételt!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: van∈L
Ef: -
Uf: van=∃i∈[e..u]:(T(i))
Rövidítve:
Uf: van=VAN(i=e..u,T(i))
```

### **Algoritmus**

```
i:=e
i ≤ u és nem T(i)
i:=i+1
van:=i ≤ u
Változó
i:Egész
ván:=i
```

### Eldöntés sablon

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy T:[e..u]→Logikai feltétel. Határozzuk meg, hogy van-e az [e..u] intervallumnak olyan eleme, amely kielégíti a T feltételt!

### Specifikáció

```
Be: e∈Z, u∈Z

Ki: van∈L

Ef: -

Uf: van=∃i∈[e..u]:(T(i))

Rövidítve:
```

Uf: van=VAN(i=e..u,T(i))

### **Algoritmus**

```
van:=hamis;i:=eVáltozó<br/>i:Egésznem van és i ≤ UT(i)van:=igaz i:=i+1
```

## Példa – bukott-e visszavezetés

Egy tanuló év végi osztályzatai alapján állapítsuk meg, hogy bukott-e!

#### **Feladatsablon**

Be: e∈Z, u∈Z

Ki: van∈L

Ef: -

Uf: van=VAN(i=e..u,T(i))

### **Bukott-e**

Be: n∈N, jegyek∈N[1...n]

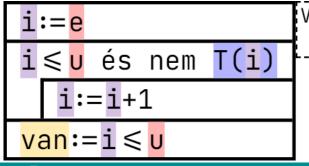
Ki: bukott∈L

Ef:  $\forall i \in [1..n]: (1 <= jegyek[i] <= 5)$ 

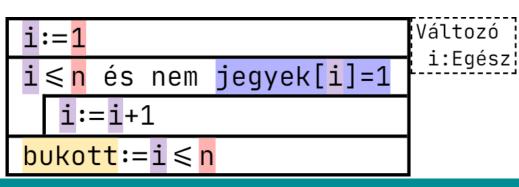
Uf: bukott=VAN(i=1..n, jegyek[i]=1)

#### Visszavezetés:

### **Algoritmus:**



```
Változó
i:Egész
```



## Kiválasztás



### Kiválasztás

### **Feladatok:**

- Ismerjük egy ember havi bevételeit és kiadásait. Év végére nőtt a vagyona. Adjunk meg egy hónapot, amikor nőtt a vagyona!
- 2. Adjuk meg egy 1-nél nagyobb természetes szám egytől különböző legkisebb osztóját!
- 3. Adjuk meg egy magyar szó egy magánhangzóját!
- 4. Adjuk meg egy hónapnévről a sorszámát!

#### Mi bennük a közös?

N "valami" közül kell megadni egy adott tulajdonságút, ha tudjuk, hogy ilyen elem biztosan van.



Ez a keresés programozási tétel olyan változata, amelyben nem kell felkészülnünk arra, hogy a keresett elemet nem találjuk meg.

# Példa – legkisebb osztó algoritmikus gondolkodással

#### **Feladat:**

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

### Specifikáció és algoritmus:

Be: n∈N

Ki: o∈N

Ef: n>1

Uf: 1<o<=n és o n és

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

```
i i<=9? i∤n?
1
e= 2 IGAZ IGAZ
3 IGAZ HAMIS
4
5
6
7
8
n= 9</pre>
```

```
i i<=5? i∤n?

1
e= 2 IGAZ IGAZ
3 IGAZ IGAZ
4 IGAZ IGAZ
n= 5 IGAZ HAMIS
```

```
i:=2

i∤n

i:=i+1

o:=i
```

### Kiválasztás sablon

#### **Feladat**

Adott egy e egész szám és egy e-től jobbra értelmezett T:Egész—Logikai feltétel. Határozzuk meg az e-től jobbra eső első olyan számot, amely kielégíti a T feltételt, ha tudjuk, hogy ilyen szám biztosan van!

### Specifikáció

```
Be: e∈Z
Ki: ind∈Z
Ef: ∃i∈[e..∞]:(T(i))
Uf: ind>=e és T(ind) és
∀i∈[e..ind-1]:(nem T(i))
Rövidítve:
Uf: ind=KIVÁLASZT(i>=e,T(i))
```

### **Algoritmus**

```
i:=e

nem T(i)

i:=i+1

ind:=i
```

```
ind:=e
nem T(ind)
ind:=ind+1
```

### Példa – magánhangzó-e visszavezetés

Adjuk meg egy magyar szó egy magánhangzóját!

### Feladatsablon

Be: e∈Z

Ki: ind∈Z

Ef:  $\exists i \in [e..\infty]:(T(i))$ 

Uf: ind=KIVÁLASZT(i>=e,T(i))

Magánhangzó

Be: szó∈S

Ki: mh∈N

Fv: magánhangzó:K->L,

magánhangzó(k)=k="a" vagy ...

Ef:  $\exists i \in [1..hossz(szó)]$ :

(magánhangzó(szó[i]))

magánhangzó(szó[i]))

T: tulajdonságfüggvény

Uf: mh=KIVÁLASZT(i>=1,

Visszavezetés:

ind ~ mh

e ~ 1

T(i) ~ magánhangzó(szó[i])

### **Algoritmus:**

ind:=e
nem T(ind)
ind:=ind+1



mh := 1

nem magánhangzó(szó[<mark>mh</mark>])

mh := mh + 1

## Másolás függvényszámítás



### Másolás

### **Feladatok:**

- Egy számsorozat tagjainak adjuk meg az abszolút értékét!
- Egy szöveget alakítsunk át csupa kisbetűssé!
- Számoljuk ki két vektor összegét!
- Készítsünk függvénytáblázatot a sin(x) függvényről!
- Ismerünk N dátumot 'éé.hh.nn' alakban, adjuk meg őket 'éé. hónapnév nn.' alakban!

#### Mi bennük a közös?

n darab "valamihez" kell hozzárendelni másik n darab "valamit", ami akár az előbbitől különböző típusú is lehet. A darabszám, a sorrend is marad.





# Példa – abszolút értékek algoritmikus gondolkodással

```
    x
    y

    1 3,3
    \rightarrow 3,3

    2 -5,8
    \rightarrow 5,8

    3 4,5
    \rightarrow 4,5

    4 -2,2
    \rightarrow 2,2
```

### **Feladat:**

Egy számsorozat tagjainak adjuk meg az abszolút értékét!

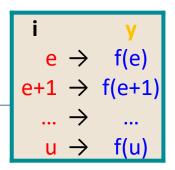
### Specifikáció:

```
Be: n∈N, x∈R[1..n]
Ki: y∈R[1..n]
Ef: -
Uf: ∀i∈[1..n]:(y[i]=abs(x[i]))
```

### **Algoritmus:**

```
i=<mark>1..n</mark>
y[i]:=abs(x[i])
Változó
i:Egész
```

### Másolás sablon



#### **Feladat**

Adott az egész számok egy [e..u] intervalluma és egy f:[e..u]→H függvény. Rendeljük az [e..u] intervallum minden értékéhez az f függvény hozzá tartozó értékét!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: y∈H[1..u-e+1]
Ef: -
Uf: ∀i∈[e..u]:(y[i-e+1]=f(i))
Rövidítve:
Uf: y=MÁSOL(i=e..u, f(i))
```

### **Algoritmus**

# Példa – két vektor összege visszavezetés

Számoljuk ki két vektor összegét!

#### Feladatsablon

(mintafeladat)

Be: e∈Z, u∈Z

Ki: y∈H[1..u-e+1]

Ef: -

Uf: y=MÁSOL(i=e..u, f(i))

### Két vektor összege

(konkrét feladat)

Be:  $n \in \mathbb{N}$ ,  $p \in \mathbb{R}[1..n]$ ,  $q \in \mathbb{R}[1..n]$ 

Ki:  $r \in R[1..n]$ 

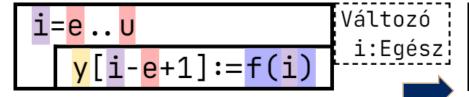
Ef: -

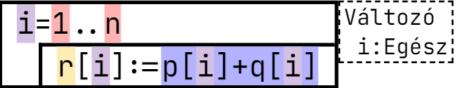
Uf: r=MÁSOL(i=1..n,p[i]+q[i])

#### Visszavezetés:

```
y ~ r
e..u ~ 1..n
f(i) ~ p[i]+q[i]
```

### **Algoritmus:**





## Kiválogatás



### Kiválogatás

### Feladatok:

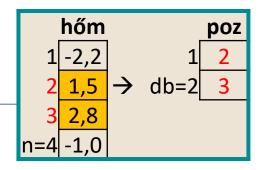
- Adjuk meg egy osztály kitűnő tanulóit!
- Adjuk meg egy természetes szám összes osztóját!
- Adjuk meg egy mondat magas hangrendű szavait!
- Adjuk meg emberek egy halmazából a 180 cm felettieket!
- Adjuk meg egy év azon napjait, amikor délben nem fagyott!
- Soroljuk föl egy szó magánhangzóit!

#### Mi bennük a közös?

n darab "valami" közül kell megadni az összes, adott T tulajdonsággal rendelkezőt!



### Példa – nem fagyos napok algoritmikus gondolkodással



#### **Feladat:**

Adjuk meg egy év azon napjait, amikor délben nem fagyott!

### **Specifikáció:**

Be:  $n \in \mathbb{N}$ ,  $h \circ m \in \mathbb{R}[1..n]$ 

Ki:  $db \in N$ ,  $poz \in N[1..db]$ 

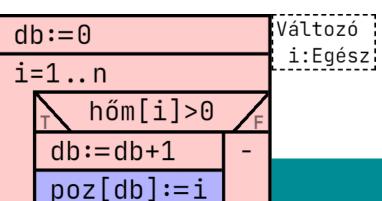
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100

Uf: db=DARAB(i=1..n,hóm[i]>0) és

 $\forall i \in [1..db]: (hőm[poz[i]]>0)$  és

 $\forall i \in [1..db]: (\forall j \in [1..db]: (i < j - poz[i] < poz[j]))$ 

### **Algoritmus:**

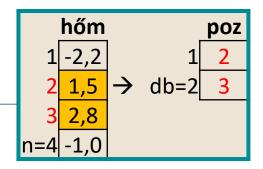


megszámolás

a kiválogatott indexekhez tartozó értékek pozitívak

nincs két egyforma index a poz tömbben, részsorozata az indextartománynak

### Példa – nem fagyos napok algoritmikus gondolkodással



#### **Feladat:**

Adjuk meg egy év azon napjait, amikor délben nem fagyott!

### Specifikáció:

Be:  $n \in \mathbb{N}$ ,  $h \circ m \in \mathbb{R}[1..n]$ 

Ki: db∈N, poz∈N[1..db]

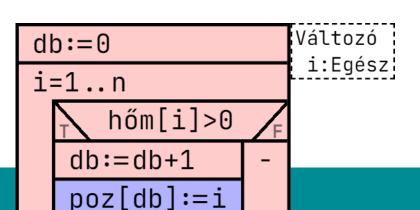
Ef: ∀i∈[1..n]:(-100<=hőm[i]<=100

Uf: db=DARAB(i=1..n,hóm[i]>0) és

∀i∈[1..db]:(hőm[poz[i]]>0) es

poz⊆[1..n]⊦

### **Algoritmus:**

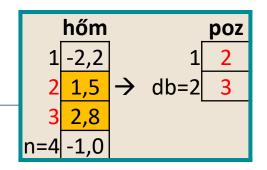


megszámolás

a kiválogatott indexekhez tartozó értékek pozitívak

nincs két egyforma index a poz tömbben

### Példa – nem fagyos napok algoritmikus gondolkodással

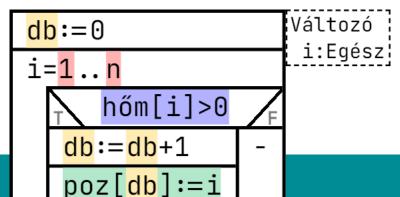


#### **Feladat:**

Adjuk meg egy év azon napjait, amikor délben nem fagyott!

### Specifikáció:

### **Algoritmus:**





### Példa – kitűnő tanulók algoritmikus gondolkodással

### 

megszámolás

#### **Feladat:**

Adjuk meg egy osztály kitűnő tanulóit!

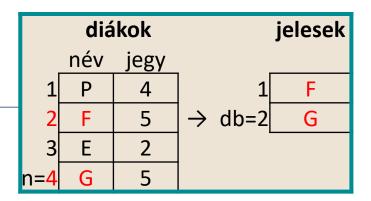
tartoznak, sőt részsorozat!

### **Specifikáció** és algoritmus:

```
Be: n∈N, diákok∈Diák[1..n], Diák=(név/S x jegy:N)
Ki: db∈N, jelesek∈S[1..db]
                                                          a kiválogatott nevek olyan
Ef: \forall i \in [1..n]: (1 < = diákok[i].jegy < = 5)
                                                         indexhez tartoznak, ahol a
Uf: [db=DARAB(i=1..n,diákok[i].jegy=5) és
                                                                 jegy 5-ös
     \forall i \in [1..db]: (\exists j \in [1..n]: (
        diákok[j].jegy=5 és jelesek[i] édiákok[j].név ) és
                                                                             Változó
     \forall i \in [1..db]: (\forall j \in [1..db]: (i < j - > j = [1..db])
                                                                              i:Egész
        \exists ii \in [1..n]: (\exists jj \in [1..n]: (
                                               db:=0
           ii≼jj és
                                               i=1..n
           dia k[ii].név=jelesek[i]
                                                          diákok[i].jegy=5
           diákok jj].név=jelesek[j]
                                                   db := db + 1
                                                  jelesek[db]:=diákok[i].név
                  a kiválogatott nevek
                 különböző indexekhez
```



### Példa – kitűnő tanulók algoritmikus gondolkodással



diákok[i].jegy=5

jelesek[db]:=diákok[i].név

megszámolás

db:=0

i=1...n

db := db + 1

#### Variáció:

Részsorozat rövidítve

### Specifikáció és algoritmus:

```
Be: n∈N, diákok∈Diák[1..n], Diák=Név x Jegy, Név=S, Jegy=N Ki: db∈N, jelesek∈S[1..db]

Ef: ∀i∈[1..n]:(1<=diákok[i].jegy<=5)

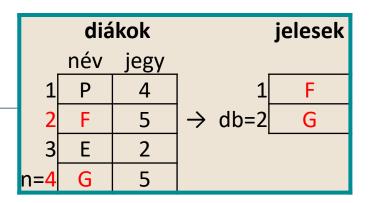
Uf: db=DARAB(i=1..n,diákok[i].jegy=5)! és jegy 5-ös

∀i∈[1..db]:(∃j∈[1..n]:(
    diákok[j].jegy=5 és jelesek[i]=diákok[j].név)) és változó i:Egész
```

a kiválogatott nevek különböző indexekhez tartoznak, sőt részsorozat!



### Példa – kitűnő tanulók algoritmikus gondolkodással



diákok[i].jegy=5

jelesek[db]:=diákok[i].név

megszámolás

i=1...n

db := db + 1

#### Variáció:

Részsorozat rövidítve

### Specifikáció és algoritmus:

```
Be: n∈N, diákok∈Diák[1..n], Diák=Név x Jegy, Név=S, Jegy=N
Ki: db∈N, jelesek∈S[1..db]
Ef: ∀i∈[1..n]:(1<=diákok[i].jegy<=5)
Uf: db=DARAB(i=1..n,diákok[i].jegy=5)! és
∀i∈[1..db]:(∃j∈[1..n]:(
    diákok[j].jegy=5 és jelesek[i]=diákok[j].név!)) és
    jegy 5-ös
∀i∈[sek⊆diákok.név

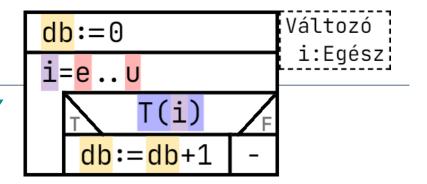
    db:=0
```

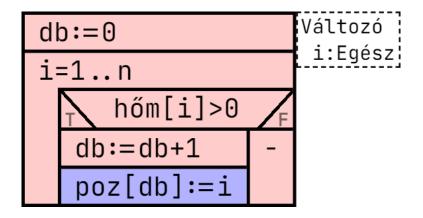
a kiválogatott nevek különböző indexekhez tartoznak



### Tanulságok

- A kiválogatás hasonlít.
  - Megszámolásra
    - Hányszor teljesült a T tulajdonság?
    - + mely esetekben?
  - Keresésre
    - Ha teljesült, akkor hol a T tulajdonság?
    - + mindenkire





```
van:=hamis; ind:=e
nem van és ind ≤ u

T(ind)
van:=igaz ind:=ind+1
```

### Tanulságok

 diákok
 jeles
 jelesek

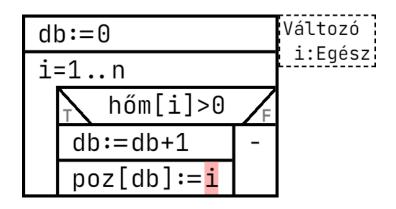
 név
 jegy
 →
 indexek
 →

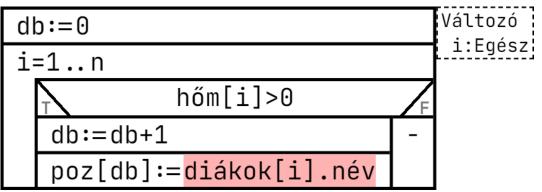
 1
 P
 4
 1
 2
 1
 F

 2
 F
 5
 db=2
 3
 db=2
 G

 3
 E
 2
 2
 n=4
 G
 5

- Eredményképpen
  - intervallum elemei, vagy
  - ezekhez rendelt értékek





# Kiválogatás sablon

```
i T(i) f(i)

e \rightarrow HAMIS

e+1 \rightarrow IGAZ \rightarrow 1 f(e+1)

e+2 \rightarrow IGAZ \rightarrow 2 f(e+2)

u \rightarrow HAMIS
```

#### **Feladat**

Adott az egész számok egy [e..u] intervalluma, egy ezen értelmezett T:[e..u]→Logikai feltétel és egy f:[e..u]→H függvény. Határozzuk meg az f függvény az [e..u] intervallum azon értékeinél felvett értékeit, amelyekre a T feltétel teljesül!

### Specifikáció

```
Be: e∈Z, u∈Z
Ki: db∈N, y∈H[1..db]
Ef: -
Uf: db=DARAB(i=e..u,T(i)) és
∀i∈[1..db]:(
∃j∈[e..u]:T(j) és y[i]=f(j))
és y⊆(f(e),f(e+1),...,f(u))
Rövidítve:
```

Uf: (db,y)=KIVÁLOGAT(i=e..u,T(i),f(i))

### **Algoritmus**



# Példa – összes osztó visszavezetés

Adjuk meg egy természetes szám összes osztóját!

#### **Feladatsablon**

(mintafeladat)

Be: e∈Z, u∈Z

Ki:  $db \in N$ ,  $y \in H[1..db]$ 

Ef: -

Uf: (db, y) =

KIVÁLOGAT(i=e..u,T(i),f(i))

### Visszavezetés:

### **Algoritmus:**

### Összes osztó

(konkrét feladat)

Be: n∈N

Ki: db∈N, osztók∈N[1..db]

Ef: -

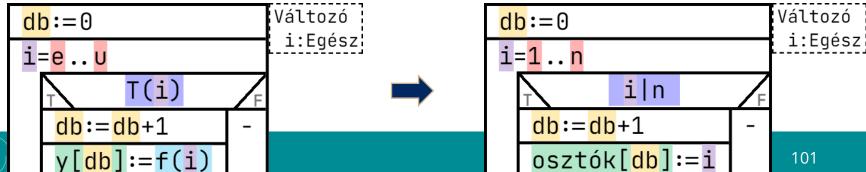
■ Uf: (db,osztók)=

~ osztók KIVÁLOGAT(i=1..n,i|n,i)

T(i) ~ i|n

e..u ~ 1..n

 $f(i) \sim i$ 





### Példa – kitűnő tanulók visszavezetés

Adjuk meg egy osztály kitűnő tanulóit!

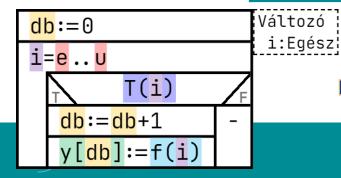
#### Feladatsablon

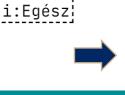
```
Be: e∈Z, u∈Z
Ki: db \in \mathbb{N}, y \in \mathbb{H}[1...db]
Ef: -
Uf: (db, y) =
       KIVÁLOGAT(i=e..u,
          T(i), y ~ jelesek
```

f(i))

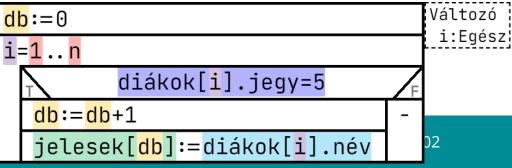
#### Kitűnő tanulók

```
Be: n∈N, diákok∈Diák[1..n],
                     Diák=(név:S x jegy:N)
                 Ki: db∈N, jelesek∈S[1..db]
                 Ef: -
                 Uf: (db,jelesek)=
                       KIVÁLOGAT(i=1..n,
                         diákok[i].jegy=5,
                           diákok[i].név)
T(i) ~ diákok[i].jegy=5
f(i) ~ diákok[i].név
```





e..u ~ 1..n



# Összefoglalás



### Feladatmegoldás lépései

### 1. Specifikáció

- a) Példa
- b) Bemenet, kimenet
  - i. egyszerű adat?
  - ii. több különböző? rekord
  - iii. több azonos? tömb
- c) Előfeltétel
- d) Utófeltétel

### 2. Algoritmus

- a) Adat <del>></del> változók
- b) Új halmazok → típusok
- c) Beolvasás
- d) Feldolgozás
  - i. támpontok az uf-ben
  - ii. végrehajtható spec.
  - iii. és, vagy, ->, ∀, ∃
  - iv. nevezetes minták
- e) Kiírás
- 3. Kód

### Analóg programozás – visszavezetés

- Visszavezetés
  - Konkrét feladat felírása
  - Összevetés a minta sablonjával
  - Különbségek felírása egy táblázatba
  - Különbségek alkalmazása a sablon algoritmusában
  - → Konkrét feladat algoritmusa

## Programozási minták

1.	Összegzés	
2.	Megszámolás	szummás, mindenes feladat
3.	Maximumkiválasztás	számlálós ciklus
4.	Feltételes maximumkeresés	
5.	Keresés	létezikes feladat
6.	Eldöntés	feltételes ciklus
7.	Kiválasztás	Jeneteies cikius
8.	Másolás	szummás, mindenes feladat
9.	Kiválogatás	↓ számlálós ciklus

## Programozási minták

1.	Összegzés	üres intervallumra is
2.	Megszámolás	ures intervalialina is
3.	Maximumkiválasztás	legalább 1 elemű intervallum
4.	Feltételes maximumkeresés	
5.	Keresés	üres intervallumra is
6.	Eldöntés	
7.	Kiválasztás	legalább 1 elemű intervallum
8.	Másolás	
9.	Kiválogatás	üres intervallumra is



### Programozási minták

- 1. Összegzés
- 2. Megszámolás
- 3. Maximumkiválasztás
- 4. Feltételes maximumkeresés

intervallumhoz egy értéket rendel (néha kettő, három értéket)

- 5. Keresés
- 6. Eldöntés
- 7. Kiválasztás
- 8. Másolás

9. Kiválogatás

intervallumhoz értéksorozatot rendel (tömböt)

## Ellenőrző kérdések



### Ellenőrző kérdések

- 1. Mik a visszavezetés lépései?
- 2. Add meg az egyes programozási minták specifikációit!
- 3. Add meg az egyes programozási minták algoritmusait!