



ELTE | IK

PROGRAMOZÁS

Programozási minták általánosítása

Horváth Győző

Ismétlés



Programozási minták

1. Összegzés
2. Megszámolás
3. Maximumkiválasztás
 - a. Minimumkiválasztás
4. Feltételes maximumkeresés
5. Keresés
6. Eldöntés
 - a. Mind eldöntés
7. Kiválasztás
8. Másolás
9. Kiválogatás

Most Common DUPLO Parts



Brick Architect 



Összegzés általánosítása „játék” a programozási mintákkal



Összegzés

Feladatok:

1. 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 szakkör adatait. Adjuk meg a szakkörönként. Adjuk meg **Mi bennük a közös?** n szám összegét kell kiszámolni!
5. Ismerünk N szót. Adjuk meg a belőlük összeállított mondatot!

Összegzés

Feladatok:

1. 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 szakköreire járó tanulóit, szakkörönként. Adjuk meg, kik járnak szakkörre!
5. Ismerünk N szót. Adjuk meg a belőlük összeállított mondatot!

Összegzés

Mi bennük a közös?

n „valami” szorzatát, unióját, összefűzését kell kiszámolni!
Ugyanúgy, mint összegzés esetén, de ebben az esetben összeadás helyett más műveletet kell alkalmazni az egyes „valamik” között.

Feladatok:

1. Adjuk meg az n számhoz az n **faktoriális** értékét!

$$f = 1 * 2 * 3 * \dots * n = \prod_{i=1..n} i$$

2. Ismerjük egy iskola szakköreire járó tanulóit, szakkörönként.
Adjuk meg, kik járnak szakkörre!

$$\text{szakkörösök} = \emptyset \cup \text{nevek1} \cup \text{nevek2} \dots \cup \text{nevekn} = \bigcup_{i=1..n} \text{nevek}[i]$$

3. Ismerünk N szót. Adjuk meg a belőlük összeállított mondatot!

$$\text{mondat} = "" + \text{szó1} + \text{szó2} + \text{szó3} + \dots = \bigoplus_{i=1..n} \text{szó}[i]$$

Összegzés

Közös tulajdonságok:

- zárt intervallum (e..u)
- intervallum elemeihez rendelt érték ($f(i)$)
- **számítási művelet**, amivel az értékeket összesíteni lehetett (+, *, \cup , összefűzés)
 - Ehhez a művelethez mindig tartozik egy olyan érték, amelyikkel bármilyen másik értékkel elvégezve a műveletet, a másik értéket kapjuk. = **nullelem**
 - Pl. $0+s=s$, $1*s=s$, $\emptyset \cup s=s$, stb.
 - asszociatív művelet

A számítási művelet operandusai ugyanabból a H halmazból kerülnek ki.

Példa

Adjuk meg az n számhoz az n faktoriális értékét

Feladatsablon

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

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

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i))$

Uf: $s = \text{SZUMMA}(i = e..u, f(i), 0, +)$

n faktoriális

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

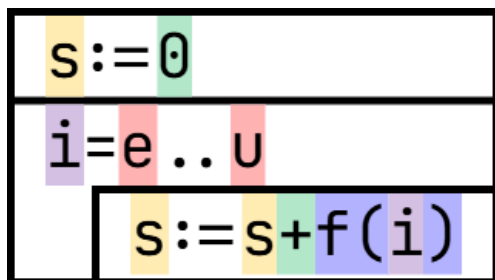
Ki: $f \in \mathbb{N}$

Ef: -

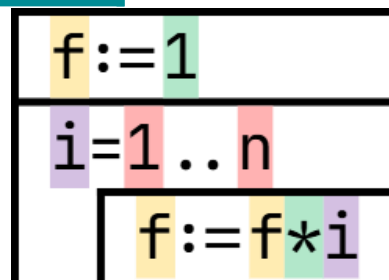
Uf: $f = n! = 1 * 2 * 3 * \dots * n = \prod (i = 1..n, i)$

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

s	\sim	f
$e..u$	\sim	$1..n$
$f(i)$	\sim	i
$0, +$	\sim	$1, *$



Változó
 $i: \text{Egész}$



Változó
 $i: \text{Egész}$

Általános összegzés sablon

Művelet neve	Operátor	Nul	Művelet neve	Operátor	Nullelem
összeadás	+	0	unió	U	\emptyset
szorzás	*	1	logikai és	és	igaz
szövegösszefűzés	+	""	logikai vagy	vagy	hamis
			tömbösszefűzés	hozzáfűz	üres tömb

Feladat

Adott az egész számok egy $[e..u]$ intervalluma és egy $f:[e..u] \rightarrow H$ függvény. A H halmaz elemein értelmezett egy asszociatív, baloldali nulla elemmel rendelkező művelet, amit most összeadásnak nevezünk és $+$ -szal jelöljük. 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

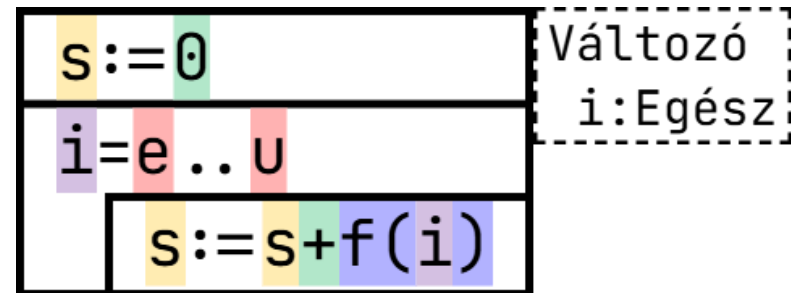
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$

Algoritmus



Általános összegzés megszámolás

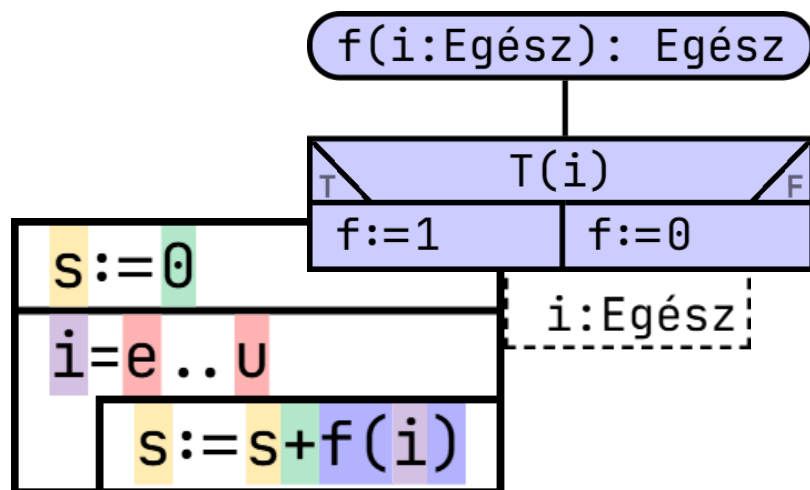
Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$



Megszámolás

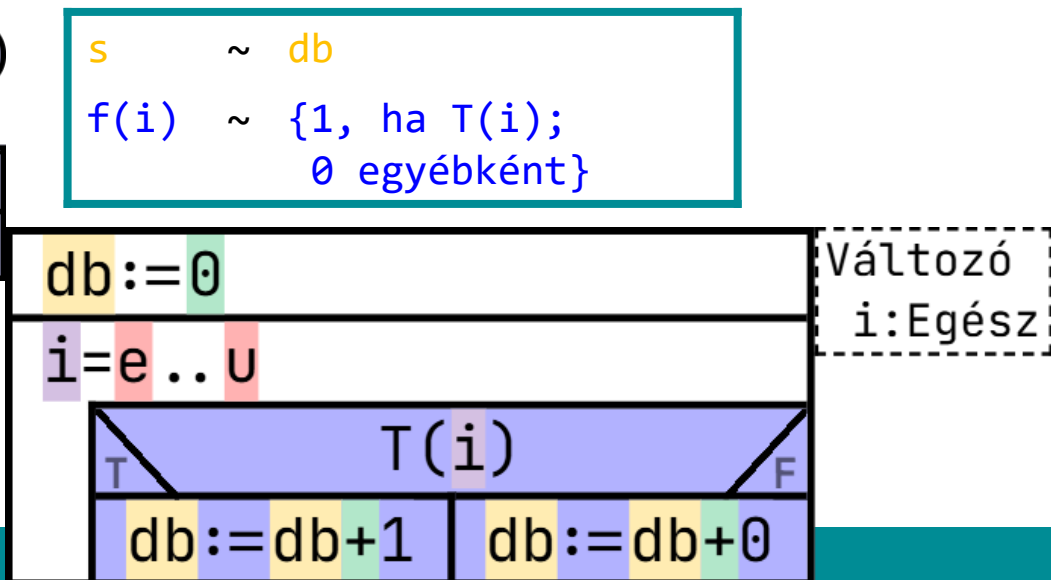
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $f: \mathbb{N} \rightarrow \mathbb{N}$, $f(i) = \{1, \text{ ha } T(i);$
 $0 \text{ egyébként}\}$

Ef: -

Uf: $db = \text{SZUMMA}(i=e..u, f(i), 0, +)$



Általános összegzés

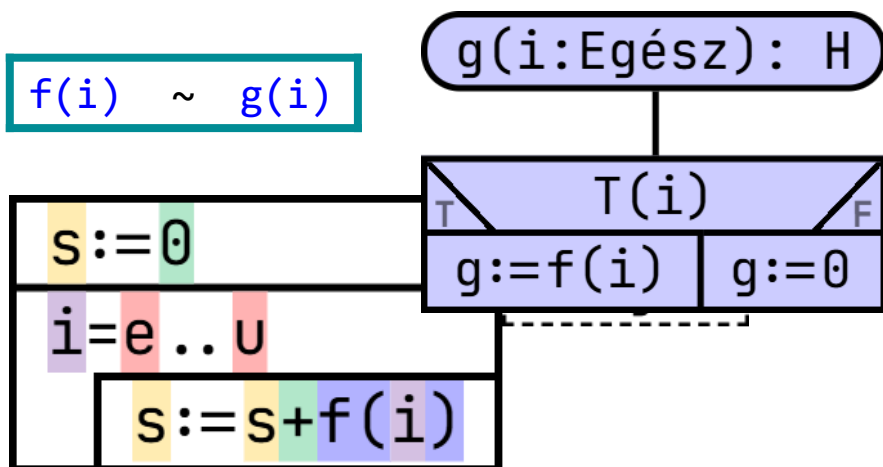
Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$



Feltételes összegzés

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

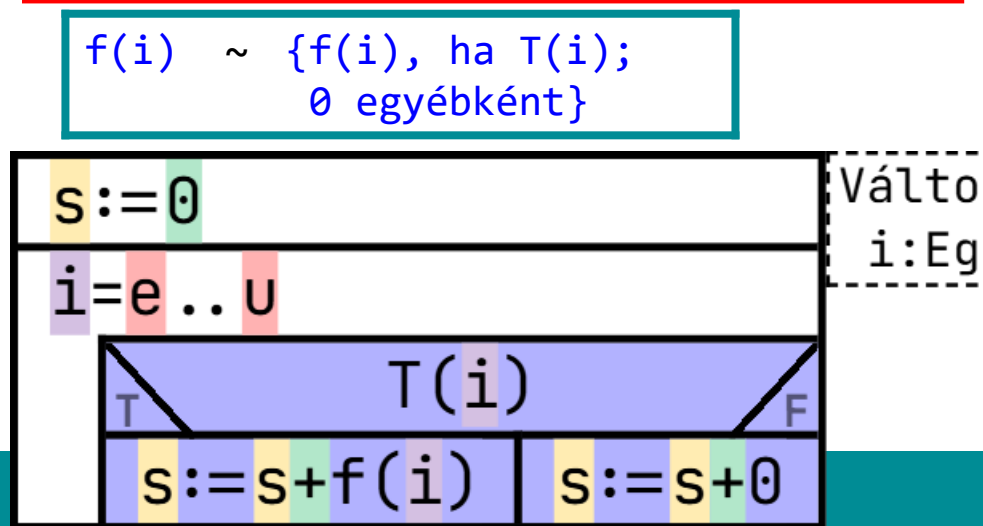
Ki: $s \in H$

Fv: $g: \mathbb{N} \rightarrow H$, $g(i) = \{f(i), \text{ ha } T(i);$
 $\emptyset \text{ egyébként}\}$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, g(i), 0, +)$

Uf: $s = \text{SZUMMA}(i=e..u, g(i), T(i), 0, +)$



Általános összegzés másolás

Feladatsablon

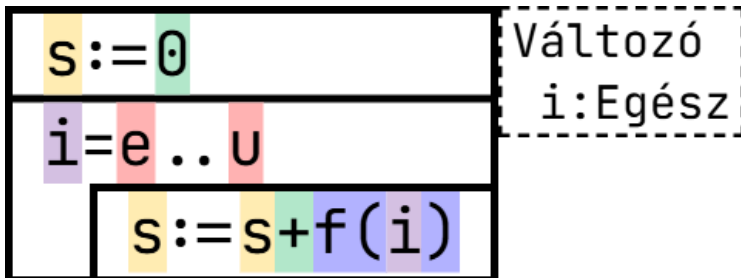
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$

s	\sim	y
$f(i)$	\sim	$[f(i)]$
$0, +$	\sim	üres tömb, hozzáfűz



Másolás

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $y \in H[1..u-e+1]$

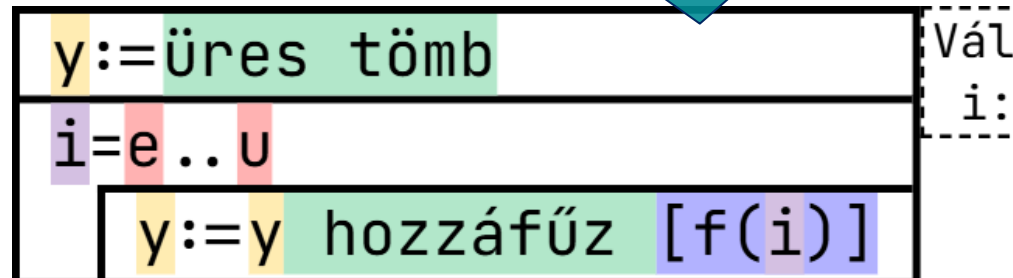
Fv: $g: \mathbb{Z} \rightarrow H[]$, $g(i) = [f(i)]$

Ef: -

Uf: $y = \text{SZUMMA}(i=e..u, g(i),$
 $\text{üres tömb, hozzáfűz})$

Pontosabban:
üres sorozat

Dinamikus tömbnél jól használható!



Általános összegzés kiválogatás

Két speciális feltételes összegzés:
megszámolás és tömbösszefűzés

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$

$s \sim db$
 $f(i) \sim \{1, \text{ ha } T(i);$
 $\quad \quad \quad 0 \text{ egyébként}\}$
 $0, + \sim 0, +$

$s \sim y$
 $f(i) \sim \{[f(i)], \text{ ha } T(i);$
 $\quad \quad \quad [] \text{ egyébként}\}$
 $0, + \sim \text{üres tömb, hozzáfűz}$

Kiválogatás

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $g: \mathbb{N} \rightarrow \mathbb{N}$, $g(i) = \{1, \text{ ha } T(i);$
 $\quad \quad \quad 0 \text{ egyébként}\}$

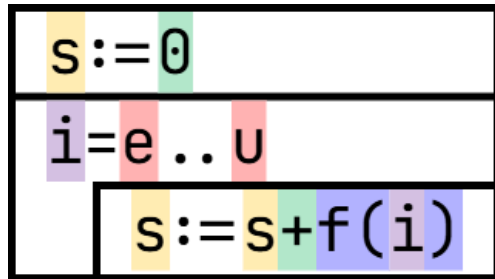
Fv: $h: \mathbb{N} \rightarrow H[]$, $h(i) = \{[f(i)], \text{ ha } T(i);$
 $\quad \quad \quad [] \text{ egyébként}\}$

Ef: -

Uf: $db = \text{SZUMMA}(i=e..u, g(i), 0, +)$ és
 $y = \text{SZUMMA}(i=e..u, h(i),$
 $\quad \quad \quad \text{üres tömb, hozzáfűz})$

Általános összegzés kiválogatás

Feladatsablon



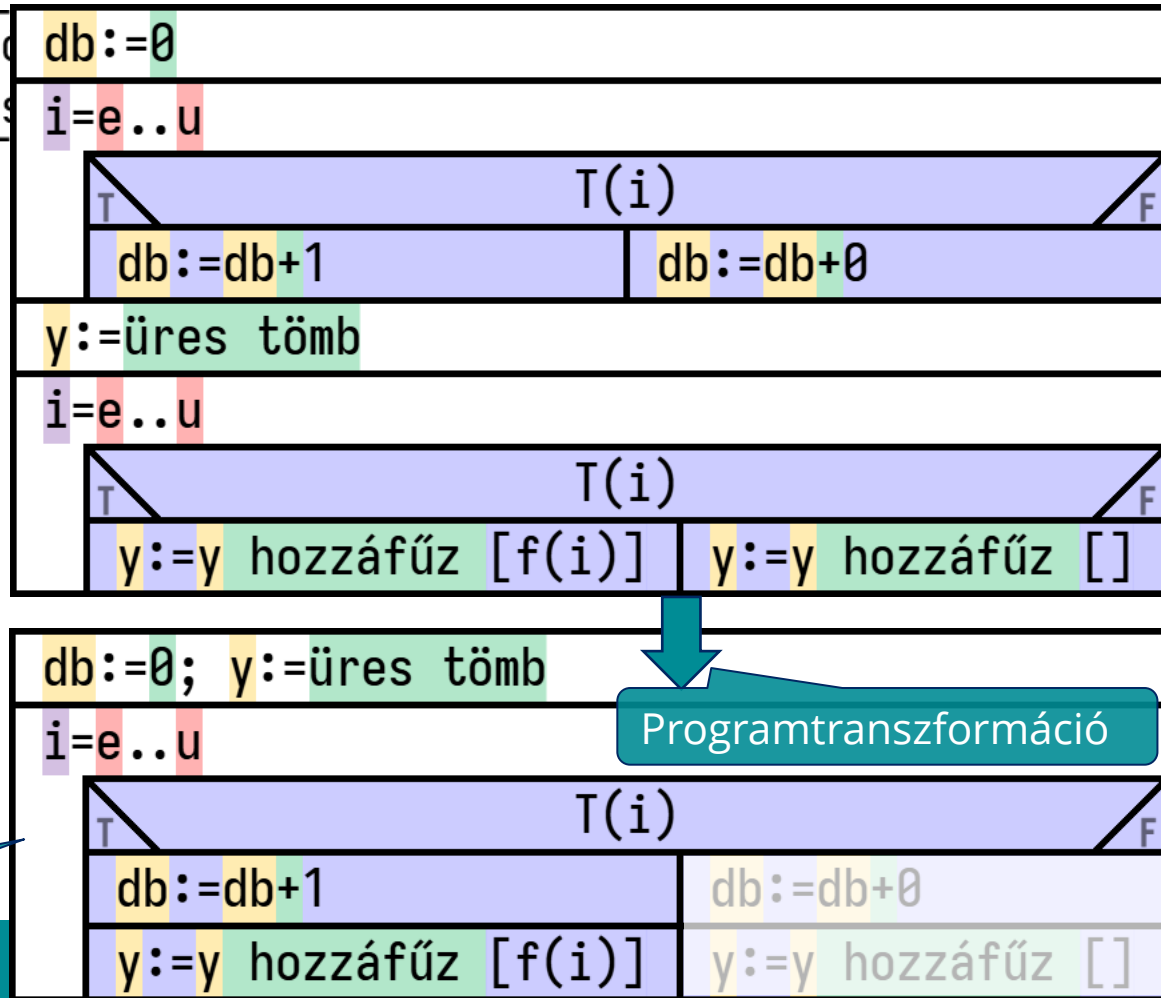
Változók
 i : Egész

$s \sim y$
 $f(i) \sim \{[f(i)], \text{ ha } T(i);$
 $\quad \quad \quad [] \text{ egyébként}\}$
 $0, + \sim \text{üres tömb, hozzáfűz}$

$s \sim db$
 $f(i) \sim \{1, \text{ ha } T(i);$
 $\quad \quad \quad 0 \text{ egyébként}\}$
 $0, + \sim 0, +$

Dinamikus tömbnél
jól használható!

Kiválogatás



Általános összegzés kiválogatás

Feladatsablon

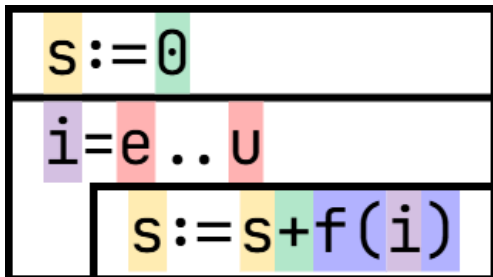
Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

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

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), 0, +)$

$s \sim y$
 $f(i) \sim \{[f(i)], \text{ ha } T(i);$
 $\quad \quad \quad [] \text{ egyébként}\}$
 $0, + \sim \text{üres tömb, hozzáfűz}$



Változó
 $i: \text{Egész}$

Kiválogatás

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

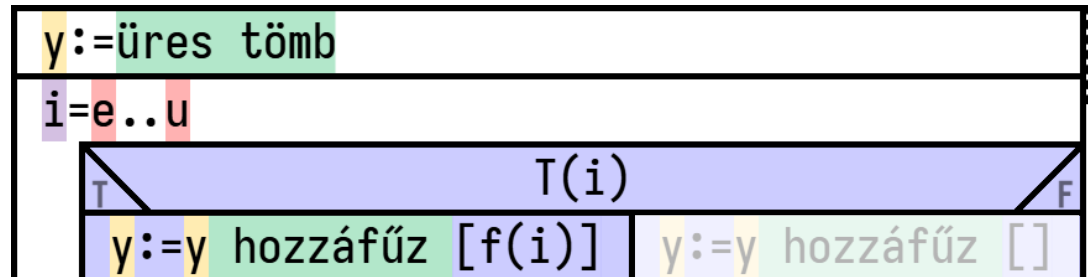
Ki: $y \in \mathbb{H}[1..]$

Fv: $h: \mathbb{N} \rightarrow \mathbb{H}[], h(i) = \{[f(i)], \text{ ha } T(i);$
 $\quad \quad \quad [] \text{ egyébként}\}$

Ef: -

Uf: $y = \text{SZUMMA}(i=e..u, h(i),$
 $\quad \quad \quad \text{üres tömb, hozzáfűz})$

Elhagyva a darabszámot



Programozási minták

1. Összegzés
 - a. Megszámolás
 - b. (Feltételes összegzés)
 - c. Másolás
 - d. Kiválogatás
2. Maximumkiválasztás
 - a. Minimumkiválasztás
3. Feltételes maximumkeresés
4. Keresés
 - a. Eldöntés
 - b. Mind eldöntés
5. Kiválasztás

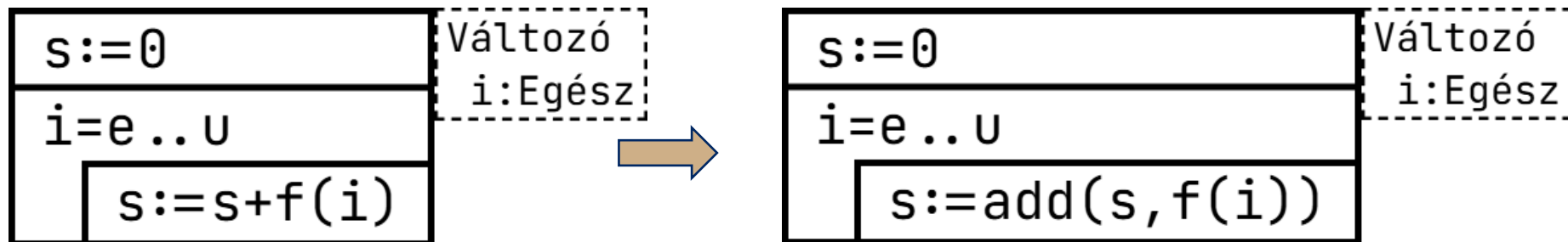
Most Common DUPLO Parts



Még általánosabb összegzés felé

A + műveletet tekintsük egy kétoperandusú függvénynek!

- $a + b \rightarrow +(a,b) \rightarrow \text{add}(a,b)$
- előnye, hogy a és b típusa eltérő lehet!



Még általánosabb összegzés felé

Ötlet

Próbáljuk meg valahogyan másképpen kifejezni azt, hogy a részösszeg aktuális értéke az előző részösszeg és az aktuális érték „összege” → rekurzív (önhivatkozó) felírás

Specifikáció

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

Ki: $s \in G$

Fv: $f: \mathbb{Z} \rightarrow H$

Ef: -

Uf: $s = kezd + f(e) + f(e+1) + f(e+2) + \dots + f(u)$

Uf: $s = \text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

Uf: $s = \text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

$\text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

$\text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

$\text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

$\text{add}(\text{add}(\text{add}(\text{add}(\text{add}(kezd, f(e)), f(e+1)), f(e+2)), \dots), f(u))$

$SZUMMA(e, u) = \{\text{add}(SZUMMA(e, u-1), f(u)), \text{ ha } e \leq u;$
 $\text{kezd} \text{ egyébként} \}$

Az add művelet asszociativitása teszi lehetővé ezt az átírást..

Még általánosabb összegzés sablon

Feladat

Adott az egész számok egy $[e..u]$ intervalluma, egy $f:[e..u] \rightarrow H$ függvény, egy $\text{add}: G \times H \rightarrow G$ függvény és egy G -beli kezdőérték. A kezdőértékből kiindulva szeretnénk egy kumulált értéket meghatározni az f függvény $[e..u]$ intervallumon felvett értékein sorban alkalmazva az add függvényt.

Specifikáció

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $\text{kezd} \in G$

Ki: $s \in G$

Fv: $f: \mathbb{Z} \rightarrow H$

Fv: $\text{add}: G \times H \rightarrow G$

Fv: $\text{SZUMMA}: \mathbb{Z} \times \mathbb{Z} \rightarrow G$,

$\text{SZUMMA}(e, u) = \{\text{add}(\text{SZUMMA}(e, u-1), f(u)), \text{ ha } e \leq u;$
 $\text{kezd egyébként}\}$

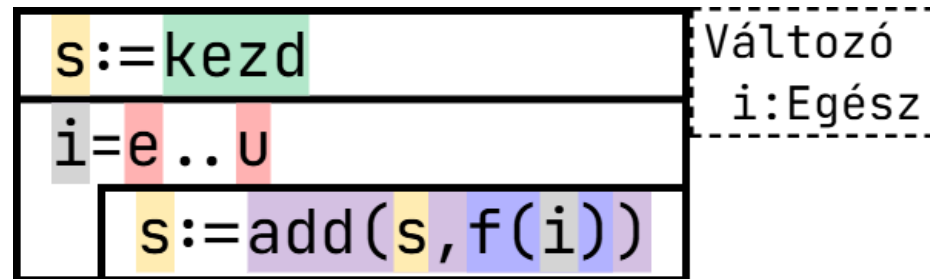
Ef: -

Uf: $s = \text{SZUMMA}(e, u)$

Rövidítve:

Uf: $s = \text{SZUMMA}(i=e..u, f(i), \text{kezd}, \text{add})$

Algoritmus



$$s = \sum_{i=e}^u f(i) = \sum_{i=e}^{u-1} f(i) + f(u)$$

Még általánosabb összegzés sablon

Specifikáció

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

Ki: $s \in G$

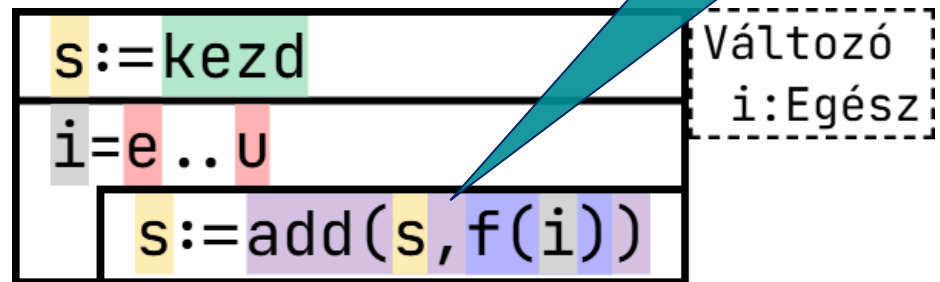
Fv: $f: \mathbb{Z} \rightarrow H$

Fv: $add: G \times H \rightarrow G$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), kezd, add)$

Algoritmus



Feladat	kezd	add(s,p)
Összegzés	0	$add(s,p) = s + p$
Produktum	1	$add(s,p) = s * p$
Maximumkiválasztás	$-\infty$ vagy $f(e)$	$add(s,p) = \max(s,p)$
Másolás	[] (üres tömb)	$add(s,p) = \text{Végére}(s,p)$
Megszámolás	0	$add(s,p) = \{s+1, \text{ ha } p;$ $s \text{ egyébként}\}$
Kiválogatás	[] (üres tömb)	$add(s,p) = \{\text{Végére}(s,p), \text{ ha } T(p);$ $s \text{ egyébként}\}$
Eldöntés	hamis	$add(s,p) = s \text{ vagy } p$

Még általánosabb összegzés

maximumkiválasztás

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

Ki: $s \in G$

Fv: $add: G \times H \rightarrow G$

Ef: -

Uf: $s =$

$SZUMMA(i=e..u, f(i), kezd, add)$

Maximumkiválasztás

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $maxért \in H$

Fv: $add: H \times H \rightarrow H$, $add(s, p) = \max(s, p)$

Ef: -

Uf: $maxért =$

$SZUMMA(i=e..u, f(i), f(e), add)$

s	\sim	$maxért$
$kezd$	\sim	$f(e)$
$add(s, p)$	\sim	$\max(s, p)$

$s := kezd$
$i = e .. u$
$s := add(s, f(i))$

Változó
 $i: \text{Egész}$

$maxért := f(e)$
$i = e .. u$
$maxért := \max(maxért, f(i))$

Változó
 $i: \text{Egész}$

Még általánosabb összegzés megszámolás

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

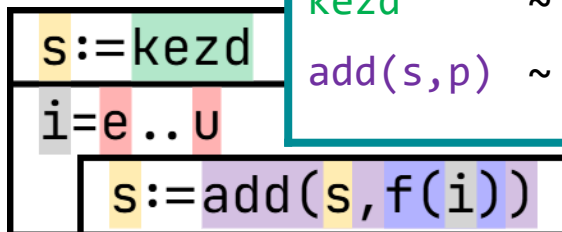
Ki: $s \in G$

Fv: $\text{add}: G \times H \rightarrow G$

Ef: -

Uf: $s = \text{SZUMMA}(i=e..u, f(i), kezd, \text{add})$

s	$\sim db$
$f(i)$	$\sim T(i)$
$kezd$	~ 0
$\text{add}(s, p)$	$\sim \{s+1, \text{ ha } p;$ $s \text{ egyébként}\}$



Megszámolás

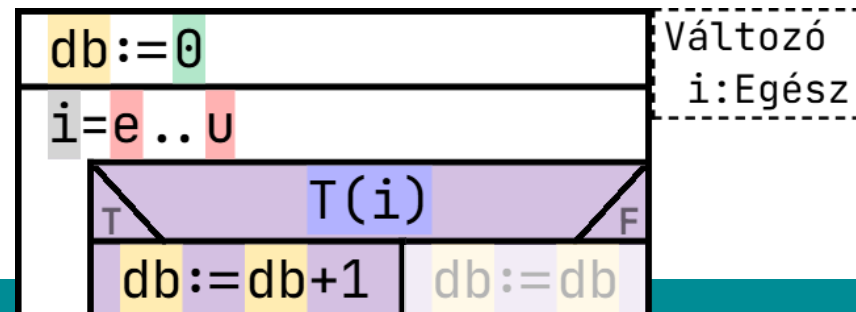
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $\text{add}: \mathbb{N} \times \mathbb{L} \rightarrow \mathbb{N}$,
 $\text{add}(s, p) = \{s+1, \text{ ha } p;$
 $s \text{ egyébként}\}$

Ef: -

Uf: $db = \text{SZUMMA}(i=e..u, T(i), 0, \text{add})$



Még általánosabb összegzés eldöntés

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

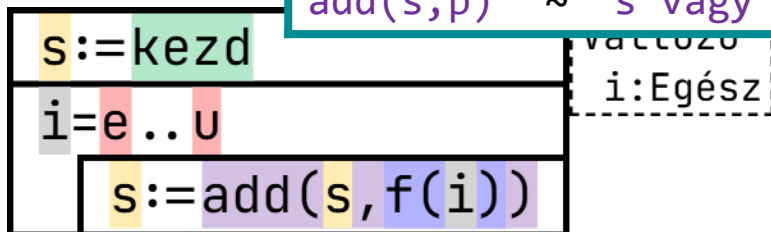
Ki: $s \in G$

Fv: $\text{add}: G \times H \rightarrow G$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i), \text{kezd}, \text{add})$

s	\sim	van
$f(i)$	\sim	$T(i)$
kezd	\sim	hamis
$\text{add}(s, p)$	\sim	$s \text{ vagy } p$



Eldöntés

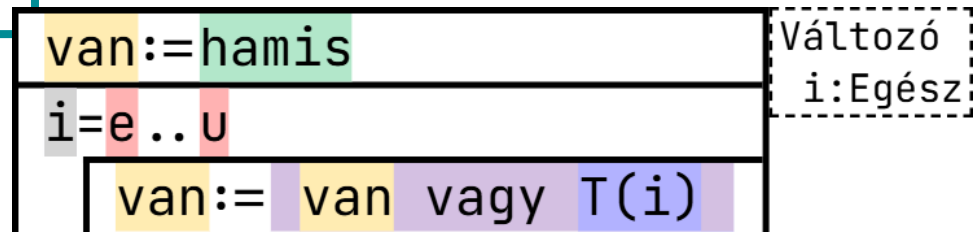
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $\text{add}: L \times L \rightarrow L$,
 $\text{add}(s, p) = s \text{ vagy } p$

Ef: -

Uf: $\text{db} = \text{SZUMMA}(i = e..u, T(i), \text{hamis}, \text{add})$



Még általánosabb összegzés kiválogatás

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

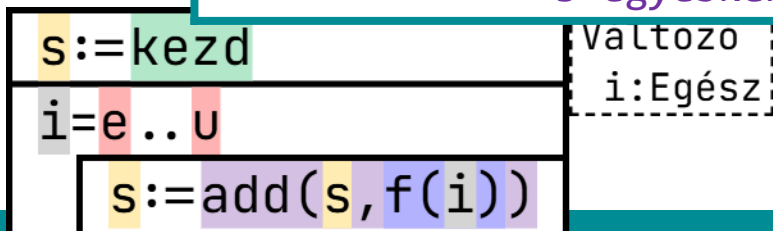
Ki: $s \in G$

Fv: $\text{add}: G \times H \rightarrow G$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i), kezd, \text{add})$

s	\sim	y
$kezd$	\sim	$[]$
$\text{add}(s, p)$	\sim	$\{\text{Végére}(s, p), \text{ ha } T(p);$ $s \text{ egyébként}\}$



Kiválogatás

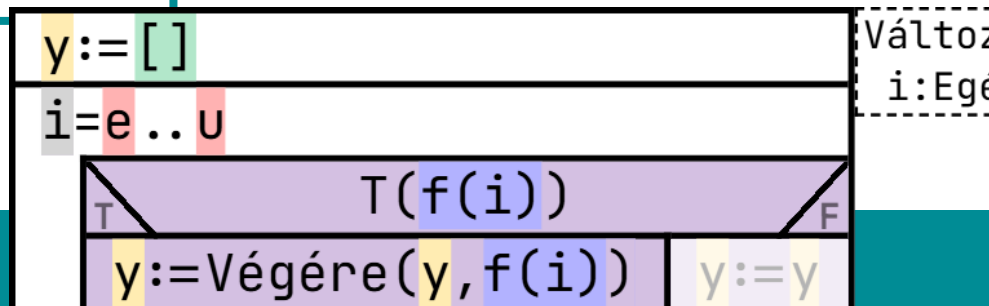
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $\text{add}: H[] \times H \rightarrow H[]$,
 $\text{add}(s, p) = \{\text{Végére}(s, p), \text{ ha } T(p);$
 $s \text{ egyébként}\}$

Ef: -

Uf: $y = \text{SZUMMA}(i = e..u, f(i), [], \text{add})$



Még általánosabb összegzés kiválogatás

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$, $kezd \in G$

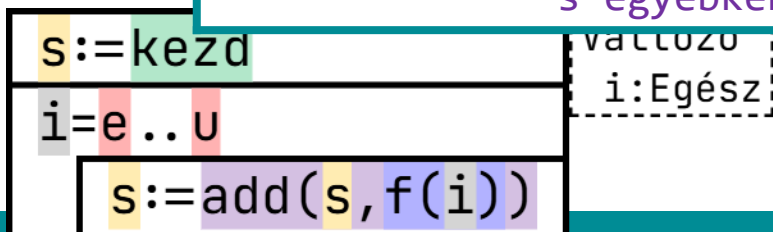
Ki: $s \in G$

Fv: $\text{add}: G \times H \rightarrow G$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i), \text{kezd}, \text{add})$

s	\sim	y
kezd	\sim	$[]$
$\text{add}(s, p)$	\sim	$\{\text{Végére}(s, p), \text{ ha } T(i);$ $s \text{ egyébként}\}$



Kiválogatás

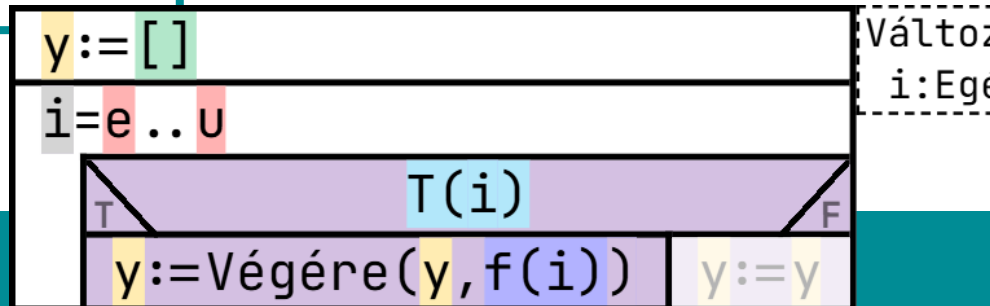
Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

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

Fv: $\text{add}: H[] \times H \times \mathbb{Z} \rightarrow H[]$,
 $\text{add}(s, p, i) = \{\text{Végére}(s, p), \text{ ha } T(i);$
 $s \text{ egyébként}\}$

Ef: -

Uf: $y = \text{SZUMMA}(i = e..u, f(i), [], \text{add})$



Programozási minták

1. Általános összegzés
 - a. Megszámolás
 - b. Feltételes összegzés
 - c. Másolás
 - d. Kiválogatás
 - e. Maximumkiválasztás
 - f. Minimumkiválasztás
 - g. Feltételes maximumkeresés
 - h. Keresés
 - i. Eldöntés
 - j. Mind eldöntés
 - k. Kiválasztás

Most Common DUPLO Parts



„Programozási tételek”



Programozási tételek

- A programozási mintákat sokszor hívják **programozási tétel**nek is
- Tágabb értelemben szinonimaként
 - „A visszavezetés során mintaként használt *specifikáció-algoritmus* párt **programozási tétel**nek hívjuk. Az elnevezés onnan származik, hogy egy minta specifikáció-algoritmus párt egy matematikai tételhez hasonlóan alkalmazunk: ha egy kitűzött feladat specifikációja hasonlít a mintafeladat specifikációjára, akkor a mintafeladat algoritmusá lényegében megoldja a kitűzött feladatot is.”
- Szűkebb értelemben
 - A tömbökre kimondott mintafeladatok a programozási tételek

Példa összegzés

Intervallum

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i = e..u, f(i))$

Tömb

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

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

Ef: -

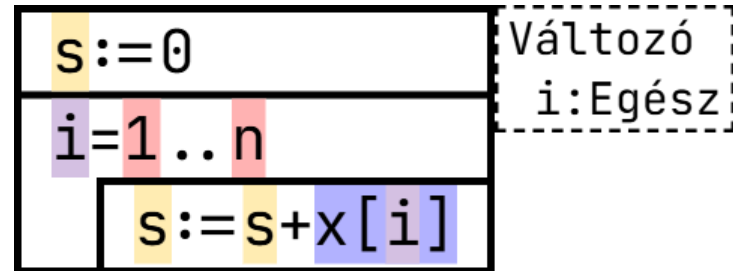
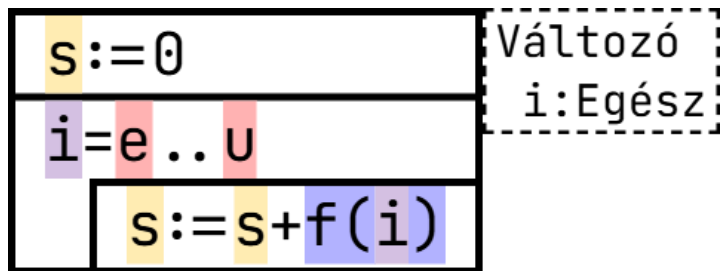
Uf: $s = \text{SZUMMA}(i = 1..n, x[i])$



Visszavezetés:

$e..u$	\sim	$1..n$
$f(i)$	\sim	$x[i]$

Algoritmus:



Példa megszámolás

Intervallum

Be: $e \in \mathbb{Z}, u \in \mathbb{Z}$

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

Ef: -

Uf: $db = \text{DARAB}(i=e..u, T(i))$

Tömb

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

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

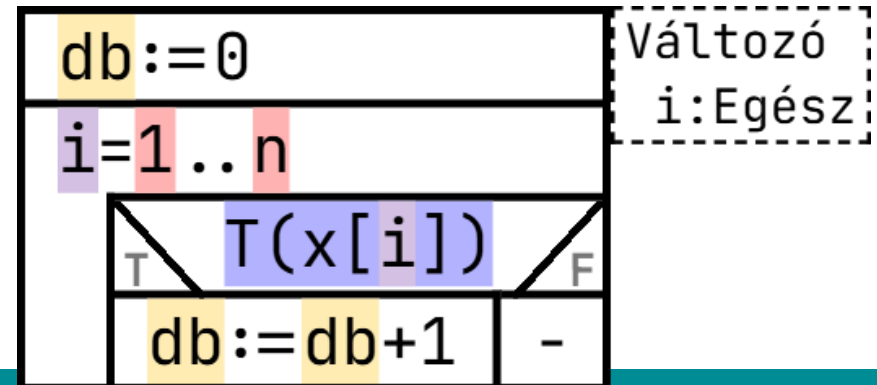
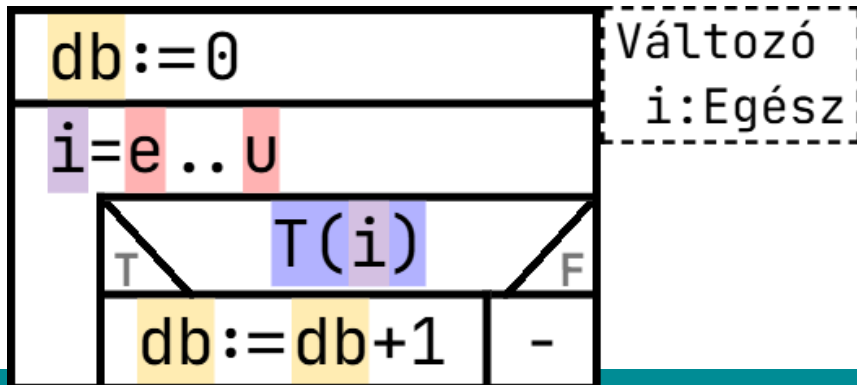
Ef: -

Uf: $db = \text{DARAB}(i=1..n, T(x[i]))$

Visszavezetés:

$e..u \sim 1..n$
 $T(i) \sim T(x[i])$

Algoritmus:



Minta \rightarrow Tétel

Sablon		Tétel
$f(i)$	\rightarrow	$x[i]$
$T(i)$	\rightarrow	$T(x[i])$

Összegzés programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb. A H halmaz elemein értelmezett az összeadás művelet. Határozzuk meg a tömb elemeinek az **összegét**, azaz a $\sum_{i=1}^n x[i]$ kifejezés értékét!

Specifikáció

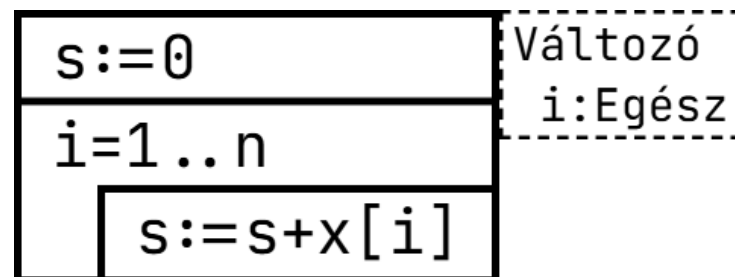
Be: $n \in \mathbb{N}$, $x \in H[1..n]$

Ki: $s \in H$

Ef: -

Uf: $s = \text{SZUMMA}(i=1..n, x[i])$

Algoritmus



Megszámolás programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai feltétel}$. Határozzuk meg, hogy a tömb elemeire a T feltétel **hányszor** veszi fel az igaz értéket!

Specifikáció

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

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

Ef: -

Uf: $db = \text{SZUMMA}(i=e..u, 1, T(x[i]))$

Rövidítve:

Uf: $db = \text{DARAB}(i=e..u, T(x[i]))$

Algoritmus

db:=0		Változó i:Egész					
i=1..n							
<table><tr><td>T</td><td>T(x[i])</td><td>F</td></tr><tr><td colspan="2">db:=db+1</td><td>-</td></tr></table>			T	T(x[i])	F	db:=db+1	
T	T(x[i])	F					
db:=db+1		-					

Maximumkiválasztás programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó, nem üres x tömb. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg a tömb legnagyobb elemének **értékét** és **indexét**!

Specifikáció

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

Ki: $\text{maxind} \in \mathbb{Z}$, $\text{maxért} \in H$

Ef: $n > 0$

Uf: $\text{maxind} \in [1..n]$ és
 $\forall i \in [1..n]: (x[\text{maxind}] \geq x[i])$ és
 $\text{maxért} = x[\text{maxind}]$

Rövidítve:

Uf: $(\text{maxind}, \text{maxért}) = \text{MAX}(i=1..n, x[i])$

Algoritmus

maxért:=x[1]; maxind:=1		Vált i:E	
i=2..n			
T	x[i]>maxért		F
maxért:=x[i]			-
maxind:=i			

Feltételes maximumkeresés programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai feltétel}$. A H halmaz elemein értelmezett egy teljes rendezési reláció. Határozzuk meg az x tömb T feltételt kielégítő elemei közül a legnagyobb elem **értékét** és **indexét**, ha egyáltalán **van** ilyen!

Specifikáció és algoritmus:

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

Ki: $\text{van} \in \mathbb{L}$, $\text{maxind} \in \mathbb{Z}$, $\text{maxért} \in H$

Ef: -

Uf: **van** = $\exists i \in [1..n]: (T(x[i]))$ és

van \rightarrow ($\text{maxind} \in [1..n]$ és

$\text{maxért} = x[\text{maxind}]$ és $T(x[\text{maxind}])$ és

$\forall i \in [1..n]: (T(x[i]) \rightarrow \text{maxért} \geq x[i])$)

Rövidítve:

Uf: (**van**, **maxind**, **maxért**) = MAX($i = 1..n$, $x[i]$, $T(x[i])$)

van:=hamis		
i=1..n		
nem T(x[i])	van és T(x[i])	nem van és T(x[i])
-	$x[i] > \text{maxért}$	van:=igaz
	maxért:=x[i]	maxért:=x[i]
	maxind:=i	maxind:=i

Keresés programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai}$ feltétel. Határozzuk meg, **hol** van az x tömb első olyan eleme, ha egyáltalán **van**, amely kielégíti a T feltételt!

Specifikáció

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

Ki: $\text{van} \in \mathbb{L}$, $\text{ind} \in \mathbb{Z}$

Ef: -

Uf: **van** $= \exists i \in [1..n] : (T(x[i]))$ és
van $\rightarrow (\text{ind} \in [1..n] \text{ és } T(x[\text{ind}]))$ és
 $\forall i \in [1..\text{ind}-1] : (\text{nem } T(x[i]))$

Rövidítve:

Uf: $(\text{van}, \text{ind}) = \text{KERES}(i=1..n, T(x[i]))$

Algoritmus

i:=1		Változó i:Egész
i≤n és nem T(x[i])		
i:=i+1		
van:=i≤n		
T	van	F
ind:=i	-	

Eldöntés programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai}$ feltétel. Határozzuk meg, hogy **van-e** a tömbnek olyan eleme, amely kielégíti a T feltételt!

Specifikáció

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

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

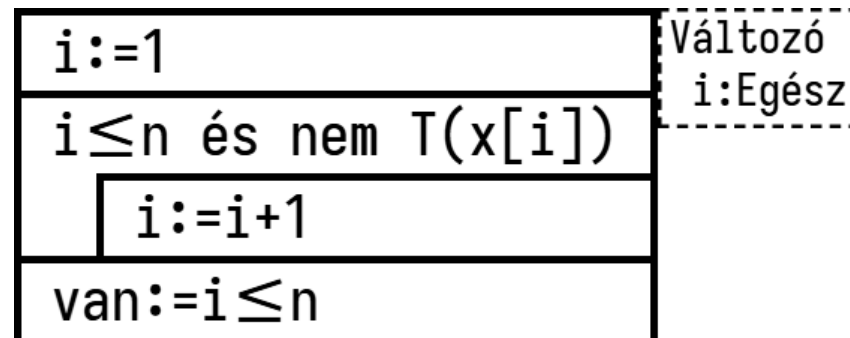
Ef: -

Uf: **van** = $\exists i \in [1..n] : (T(x[i]))$

Rövidítve:

Uf: **van** = **VAN**($i=1..n$, $T(x[i])$)

Algoritmus



Kiválasztás programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai}$ feltétel. Határozzuk meg a tömb első olyan elemének az **indexét**, amely kielégíti a T feltételt, ha tudjuk, hogy ilyen elem biztosan van!

Specifikáció

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

Ki: $\text{ind} \in \mathbb{Z}$

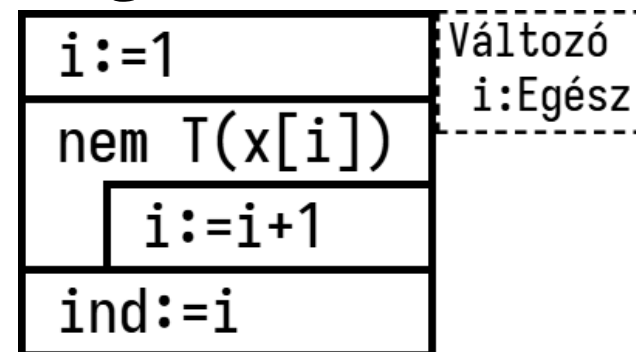
Ef: $\exists i \in [1..n] : (T(x[i]))$

Uf: **ind** \geq **1** és $T(x[\text{ind}])$ és
 $\forall i \in [1..\text{ind}-1] : (\text{nem } T(x[i]))$

Rövidítve:

Uf: **ind** = KIVÁLASZT(**i** \geq **e**, $T(x[i])$)

Algoritmus



Másolás programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $f:H \rightarrow G$ függvény. **Rendeljük** a tömb **minden eleméhez** az f függvény hozzá tartozó értékét!

Specifikáció

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

Ki: $y \in G[1..n]$

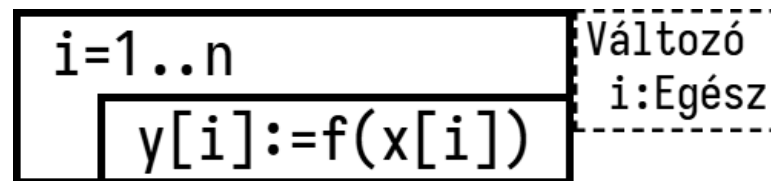
Ef: -

Uf: $\forall i \in [1..n] : (y[i] = f(x[i]))$

Rövidítve:

Uf: $y = \text{MÁSOL}(i=1..n, f(x[i]))$

Algoritmus



Kiválogatás programozási tétel

Feladat

Adott egy n elemű H halmazbeli elemeket tartalmazó x tömb és egy $T:H \rightarrow \text{Logikai feltétel}$. Határozzuk meg a tömb **azon** **elemeit**, amelyekre teljesül a T feltétel!

Specifikáció

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

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

Ef: -

Uf: $db = \text{DARAB}(i=1..n, T(x[i]))$ és
 $\forall i \in [1..db]: (T(y[i]))$ és
 $y \subseteq x$

Rövidítve:

Uf: $(db, y) = \text{KIVÁLOGAT}(i=1..n, T(x[i]), x[i])$

Algoritmus

db:=0		Változó i:Egész
i=1..n		
<div><div>T</div><div>T(x[i])</div><div>F</div></div>		
db:=db+1	-	
y[db]:=x[i]		

Programozási minták megvalósítása általánosított függvényekkel



Cél

- Programozási minták megvalósítása paraméterekkel általánosított függvényekkel
- Példa: keresés
 - Specifikáció: $(\text{van}, \text{ind}) = \text{KERES}(i = \text{e} \dots \text{u}, T(i))$
 - C# függvény: $(\text{van}, \text{ind}) = \text{Keres}(\text{e}, \text{u}, T)$
 - ahol T egy megfelelően definiált függvény
- Egy konkrét feladatnál (pl. valódi osztó keresése)
 - Specifikáció: $(\text{van}, \text{ind}) = \text{KERES}(i = 2 \dots n-1, i | n)$
 - C# függvény: $(\text{van}, \text{ind}) = \text{Keres}(2, n-1, T)$
 - ahol: `bool T(int i) { return n % i == 0; }`

Példa negatív szám

Adjuk meg, hogy hol van
negatív szám egy számokat
tartalmazó tömbben!

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{van} \in \mathbb{L}$, $\text{ind} \in \mathbb{Z}$

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$

Ef: -

Uf: $(\text{van}, \text{ind}) = \text{KERES}(i = e..u, T(i))$

Visszavezetés:

Algoritmus:

$\text{ind} \sim \text{negind}$

$e..u \sim 1..n$

$T(i) \sim x[i] < 0$

$\text{ind} := e$

$\text{ind} \leq u$ és nem $T(\text{ind})$

$\text{ind} := \text{ind} + 1$

$\text{van} := \text{ind} \leq u$

$\text{negind} := 1$

$\text{negind} \leq n$ és nem $x[\text{negind}] < 0$

$\text{negind} := \text{negind} + 1$

$\text{van} := \text{negind} \leq n$

Negatív szám

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

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

Ef: -

Uf: $(\text{van}, \text{negind}) = \text{KERES}(i = 1..n, x[i] < 0)$

Példa negatív szám

```
static (bool van, int negind) Keres
(int n, int[] x) {
    bool van; int negind;

    negind = 1;
    while (negind <= n &&
           !(x[negind - 1] < 0)) {
        negind = negind + 1;
    }
    van = negind <= n;

    return (van, negind);
}
```

Negatív szám

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

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

Ef: -

Uf: $(van, negind) = KERES(i=1..n, x[i] < 0)$

negind:=1

negind ≤ n és nem x[negind] < 0

negind:=negind+1

van:=negind ≤ n

Példa

negatív szám

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $\text{van} \in \mathbb{L}$, $\text{ind} \in \mathbb{Z}$

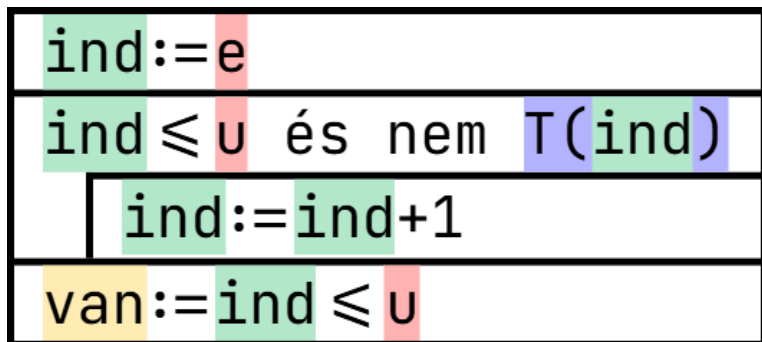
Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$

Ef: -

Uf: $(\text{van}, \text{ind}) = \text{KERES}(i = e..u, T(i))$

Visszavezetés: $\text{ind} \sim \text{negind}$

Algoritmus: $e..u \sim 1..n$



Negatív szám

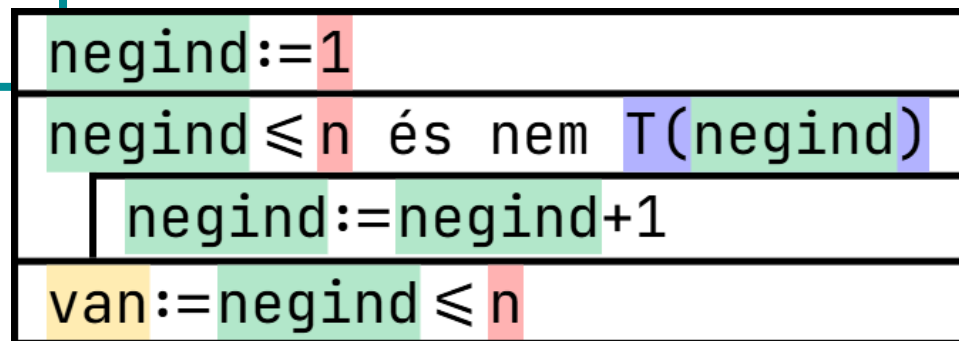
Be: $n \in \mathbb{N}$, $x \in \mathbb{Z}[1..n]$

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

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$, $T(i) = x[i] < 0$

Ef: -

Uf: $(\text{van}, \text{negind}) = \text{KERES}(i = 1..n, T(i))$



$T(i: \text{Egész}): \text{Logikai}$

$T := x[i] < 0$

Példa

negatív szám

```
static (bool van, int negind) Keres
    (int n, int[] x) {
    bool van; int negind;

    negind = 1;
    while (negind <= n && !T(negind, x)) {
        negind = negind + 1;
    }
    van = negind <= n;

    return (van, negind);
}

static bool T(int i, int[] x) {
    return x[i - 1] < 0;
}
```

Külső függvény nem lát
rá x-re, ezért át kell azt
is adni neki.

Negatív szám

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

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

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$, $T(i) = x[i] < 0$

Ef: -

Uf: $(van, negind) = KERES(i = 1..n, T(i))$

negind := 1

negind \leq n és nem T(negind)

negind := negind + 1

van := negind \leq n

$T(i: \text{Egész}): \text{Logikai}$

$T := x[i] < 0$

Példa negatív szám

1. megoldás: osztályszintű változók létrehozása
→ osztálybeli függvényekre globális

```
static int[] x;  
static (bool van, int negind) Keres  
    (int n, int[] x) {  
    bool van; int negind;  
  
    negind = 1;  
    while (negind <= n && !T(negind)) {  
        negind = negind + 1;  
    }  
    van = negind <= n;  
  
    return (van, negind);  
}  
static bool T(int i) {  
    return x[i - 1] < 0;  
}
```

Negatív szám

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

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

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$, $T(i) = x[i] < 0$

Ef: -

Uf: $(van, negind) = KERES(i = 1..n, T(i))$

negind := 1

negind ≤ n és nem T(negind)

negind := negind + 1

van := negind ≤ n

$T(i: \text{Egész}): \text{Logikai}$

$T := x[i] < 0$

Példa negatív szám

2. megoldás:

Lokális függvény: rálát a tartalmazó függvény lokális adataira, így x-et nem kell paraméterként átadni, így szignatúrája megegyezik az elvárttal.

Negatív szám

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

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

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$, $T(i) = x[i] < 0$

Ef: -

Uf: $(\text{van}, \text{negind}) = \text{KERES}(i=1..n, T(i))$

```
static (bool van, int negind) keres
(int n, int[] x) {
    bool T(int i) {
        return x[i - 1] < 0;
    }

    bool van; int negind;

    negind = 1;
    while (negind <= n && !T(negind)) {
        negind = negind + 1;
    }
    van = negind <= n;

    return (van, negind);
}
```

negind:=1

negind ≤ n és nem T(negind)

negind:=negind+1

van:=negind ≤ n

$T(i: \text{Egész}): \text{Logikai}$

$T := x[i] < 0$



Példa negatív szám

3. megoldás: Kapja meg a T függvényt
kívülről!

Függvényparaméter!

Negatív szám

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

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

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$, $T(i) = x[i] < 0$

Ef: -

Uf: $(\text{van}, \text{negind}) = \text{KERES}(i=1..n, T(i))$

```
static void Main(string[] args) {  
    Keres(n, x, T);  
}  
static (bool van, int negind) Keres  
(int n, int[] x, Func<int, int[], bool> T) {  
    bool van; int negind;  
  
    negind = 1;  
    while (negind <= n && !T(negind, x)) {  
        negind = negind + 1;  
    }  
    van = negind <= n;  
  
    return (van, negind);  
}  
static bool T(int i, int[] x) {  
    return x[i - 1] < 0;  
}
```

negind:=1

negind ≤ n és nem T(negind)

negind:=negind+1

van:=negind ≤ n

$T(i: \text{Egész}): \text{Logikai}$

$T := x[i] < 0$



A sablon megvalósítása függvényként

Feladatsablon

Be: $e \in \mathbb{Z}$, $u \in \mathbb{Z}$

Ki: $van \in \mathbb{L}$, $ind \in \mathbb{Z}$

Fv: $T: \mathbb{Z} \rightarrow \mathbb{L}$

Ef: -

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

Algoritmus:

$ind := e$
$ind \leq u$ és nem $T(ind)$
$ind := ind + 1$
$van := ind \leq u$

```
static (bool van, int ind) Keres  
(int e, int u, Func<int, bool> T) {  
    bool van; int ind;  
  
    ind = e;  
    while (ind <= u && !T(ind)) {  
        ind = ind + 1;  
    }  
    van = ind <= u;  
  
    return (van, ind);  
}
```

Az általánosított függvény használata

```
static void Main(string[] args) {  
    int n = 6; int[] x = { 1, 3, 4, -5, -2, 0 };  
    bool van; int negind;  
    bool T(int i) {  
        return x[i - 1] < 0;  
    }  
  
    (van, negind) = Keres(1, n, T);  
}  
  
static (bool van, int ind) Keres(int e, int u, Func<int, bool> T) {  
    bool van; int ind;  
  
    ind = e;  
    while (ind <= u && !T(ind)) {  
        ind = ind + 1;  
    }  
    van = ind <= u;  
  
    return (van, ind);  
}
```

Függvények lambda kifejezéssel

• $\lambda x. \lambda y. x + y$

• $\{ \lambda x. x, \lambda x. x + 1 \}$

• $\lambda x. \lambda y. x + y$

• $\lambda x. \lambda y. x + y$

(formális par.lista) => visszatérési érték

```
bool T(int i) {
    return i % 2 == 0;
}
Console.WriteLine(T(3));
```

```
Func<int, bool> T = i => i % 2 == 0;

Console.WriteLine(T(3));
```

Az általánosított függvény használata

```
static void Main(string[] args) {  
    int n = 6; int[] x = { 1, 3, 4, -5, -2, 0 };  
    bool van; int negind;  
  
    (van, negind) = Keres(1, n, i => x[i - 1] < 0);  
}  
  
static (bool van, int ind) Keres(int e, int u, Func<int, bool> T) {  
    bool van; int ind;  
  
    ind = e;  
    while (ind <= u && !T(ind)) {  
        ind = ind + 1;  
    }  
    van = ind <= u;  
  
    return (van, ind);  
}
```

Programozási minták általánosított függvényei

- Ez a fajta általánosítás mindegyik programozási minta esetén elvégezhető
- Az így kapott függvényeket egy segédosztályba helyeztük (Mintak.cs)
 - A segédosztályt vagy a feladat osztálya mellé másoljuk a Program.cs fájlban, vagy
 - fájlszinten hozzáadjuk a Mintak.cs fájlt a projekthez.
- Az általánosított függvények használata, pl:
 - `Mintak.Keres(1, n, i => x[i-1] < 0)`

Minták általánosított függvényei

összegzés

- Uf: $s = \text{SZUMMA}(i=e..u, f(i))$

- Példa:

- Uf: $s = \text{SZUMMA}(i=1..n, \text{szamok}[i])$

- Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
  
int s = Mintak.Szumma(1, n, i => szamok[i - 1]);  
int s = Mintak.Szumma(0, n - 1, i => szamok[i]);
```

- Példa:

- Uf: $s = \text{SZUMMA}(i=1..10, i)$

- Kód:

```
int s = Mintak.Szumma(1, 10, i => i);
```


Minták általánosított függvényei

általános összegzés

- Uf: $s = \text{SZUMMA}(i=e..u, f(i), \text{kezd}, \text{add})$

- Példa:

- Uf: $p = \text{SZUMMA}(i=1..n, \text{szamok}[i], 1, \text{add})$
 $\text{add}(s, p) = s * p$

- Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
  
int s = Mintak.Szumma(1, n, i => szamok[i - 1], 1, (s,p)=>s*p);
```

- Példa:

- Uf: $y = \text{SZUMMA}(i=1..n, \text{szamok}[i], [], \text{add})$
 $\text{add}(s, p) = \{\text{Végére}(s, p), \text{ ha } p < 0;$
 $\quad s \text{ egyébként}\}$

- Kód:

```
List<int> y = Mintak.Szumma(1, n, i => szamok[i - 1],  
                           new List<int>(),  
                           (s, p) => {  
                               if (p < 0) { s.Add(p); }  
                           });
```

Minták általánosított függvényei megszámolás

- Uf: $db = \text{DARAB}(i=e..u, T(i))$

- Példa:

- Uf: $db = \text{DARAB}(i=1..n, \text{szamok}[i] < 0)$

- Kód:

```
int[] szamok = { 2, 4, -7, 5, 3, -2, 1 };  
int n = szamok.Length;  
  
int db = Mintak.Darab(1, n, i => szamok[i - 1] < 0);
```

Minták általánosított függvényei

maximumkiválasztás

- Uf: $(\text{maxind}, \text{maxért}) = \text{MAX}(i=e..u, f(i))$

- Példa:

- Uf: $(\text{maxi}, \text{maxé}) = \text{MAX}(i=2..n, \text{szamok}[i] - \text{szamok}[i-1])$

- Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
  
(int maxi, int maxe) =  
    Mintak.Max(2, n, i => szamok[i - 1] - szamok[i - 2]);
```

- Példa:

- Uf: $(\text{ind}, \text{diák}) = \text{MAX}(i=1..n, \text{diákok}[i])$

- Kód:

```
struct Diak { public string nev; public int jegy; }  
  
(int ind, int diak) = Mintak.Max(1, n, i => diákok[i - 1],  
    (d1, d2) => d1.jegy > d2.jegy);
```

Minták általánosított függvényei

feltételes maximumkeresés

- Uf: $(\text{van}, \text{maxind}, \text{maxért}) = \text{FELTMAX}(\text{i}=\text{e}..u, f(\text{i}), T(\text{i}))$
- Példa:
 - Uf: $(\text{van}, \text{maxi}, \text{maxé}) = \text{FELTMAX}(\text{i}=1..n, \text{szamok}[\text{i}], \text{szamok}[\text{i}] < 0)$
 - Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };
int n = szamok.Length;

(bool van, int maxi, int maxe) =
    Mintak.Max(1, n, i => szamok[i - 1], i => szamok[i - 1] < 0);
```

Minták általánosított függvényei

keresés

- Uf: $(\text{van}, \text{ind}) = \text{KERES}(\text{i} = \text{e}.. \text{u}, \text{T}(\text{i}))$

- Példa:

- Uf: $(\text{van}, \text{ind}) = \text{KERES}(\text{i} = 1..n, \text{szamok}[\text{i}] < 0)$

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
(bool van, int ind) = Mintak.Keres(1, n, i => szamok[i - 1] < 0);
```

Minták általánosított függvényei

eldöntés

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

- Példa:

- Uf: `van=VAN(i=1..n, szamok[i]<0)`

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
bool van = Mintak.Van(1, n, i => szamok[i - 1] < 0);
```

- Példa:

- Uf: `mind=MIND(i=1..n, szamok[i]>0)`

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
bool mind = Mintak.Mind(1, n, i => szamok[i - 1] > 0);
```

Minták általánosított függvényei kiválasztás

- Uf: `ind=KIVÁLASZT(i>=e, T(i))`

- Példa:

- Uf: `ind=KIVÁLASZT(i>=1, szamok[i]<0)`

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
int ind = Mintak.Kivalaszt(1, i => szamok[i - 1] < 0);
```

Minták általánosított függvényei

másolás

- Uf: $y = \text{MÁSOL}(i = e..u, f(i))$

- Példa:

- Uf: $y = \text{MÁSOL}(i = 1..n, \text{abs}(\text{szamok}[i]))$

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
int[] y = Mintak.Masol(1, n, i => Math.Abs(szamok[i - 1]));
```


Minták általánosított függvényei

kiválogatás

- Uf: $(db, y) = \text{KIVÁLOGAT}(i = e..u, T(i), f(i))$
- Példa:
 - Uf: $(db, y) = \text{KIVÁLOGAT}(i = 1..n, \text{szamok}[i] < 0, \text{szamok}[i])$
 - Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
int[] y = Mintak.Kivalogat(1, n, i => szamok[i - 1] < 0,  
                           i => szamok[i - 1]);
```

Példa

legjobb jó tanuló

	1	2	3	m=4
1	5	5	5	3
2	5	4	4	4
3	5	3	2	4
4	5	4	5	5
n=5	5	5	4	4



Feladat:

Egy egész számokat tartalmazó mátrixban **melyik az a sor**,
amiben **csak 4-es és 5-ös van**, és az **összege a legnagyobb?**

Specifikáció:

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

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

Fv: összeg: $\mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: jó: $\mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb},) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

	1	2	3	m=4	
1	5	5	5	3	
2	5	4	4	4	17
3	5	3	2	4	→ van=igaz → legjobb=4
4	5	4	5	5	19
n=5	5	5	4	4	18

Példa

legjobb jó tanuló

	1	2	3	m=4
1	5	5	5	3
2	5	4	4	4
3	5	3	2	4
4	5	4	5	5
n=5	5	5	4	4



```
static void Main(string[] args) {
    int n; int m; int[,] jegyek;
    bool van; int legjobb;

    (n, m, jegyek) = beolvas();
    (van, legjobb, _) = Mintak.FeltMax(1, n,
        diak => osszeg(diak, m, jegyek),
        diak => jo(diak, m, jegyek)
    );
    kiir(van, legjobb);
}
```

Csak a külső szinten

Specifikáció:

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

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

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: $\text{jó}: \mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n]: (\forall \text{oszlop} \in [1..m]: (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb}, _) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

```
static void Main(string[] args) {
    int n; int m; int[,] jegyek;
    bool van; int legjobb;
```

```
    (n, m, jegyek) = beolvas();
    (van, legjobb, _) = Mintak.FeltMax(1, n,
        diak => Mintak.Szumma(1, m, tantargy => jegyek[diak - 1, tantargy - 1]),
        diak => Mintak.Mind(1, m, tantargy => 4 <= jegyek[diak - 1, tantargy - 1] &&
            jegyek[diak - 1, tantargy - 1] <= 5)
    );
    kiir(van, legjobb);
}
```

Minden szinten

Programozási tételek megvalósítása általánosított függvényekkel



Programozási tételek általánosított függvényei

- Az intervallumra kimondott programozási mintákhoz hasonló általánosítás elvégezhető a tömbökre kimondott programozási tételekre is.
- Az így kapott függvények ugyanabban a segédosztályban találhatók (Mintak.cs)
 - A segédosztályt vagy a feladat osztálya mellé másoljuk a Program.cs fájlban, vagy
 - fájl szinten hozzáadjuk a Mintak.cs fájlt a projekthez.
- Az általánosított függvényeknek ugyanaz a neve, csak más típusú paraméterekkel dolgozik (függvény túlterhelés):
 - `Mintak.Keres(x, e => e < 0)`

Programozási tételek általánosított függvényei

- Ahogy a tömbökre kimondott tételek lehetőségei szűkebbek az intervallumra kimondott mintákénál, úgy az általánosított függvényeknél is limitációkkal kell számolnunk:
 - a tömbök 0-tól indexelődnek
 - az intervallum nem határozható meg, az a tömb indextartománya
 - a tételek egyes függvényei csak az aktuális elemhez férnek hozzá, az indexhez nem (kivétel a kiválogatás)
 - mátrix helyett tömbök tömbje kell (jagged array)
 - `int[,]` \rightarrow `int[][]`

Programozási tételek általánosított függvényei

- Azért foglalkozunk a tételek nyelvi általánosításával, mert a legtöbb programozási nyelvben beépítve megtalálhatók ezek az általánosított tételek tömbfüggvények formájában.
- C#-ban ez az ún. LINQ metódusokon keresztül, a tömbök metódusaiként érhetők el.
- A saját megoldásunk mellett megmutatjuk az adott tétel LINQ megfelelőjét is.

```
using System;  
using System.Collections.Generic;  
using System.Linq;
```



Tételek általánosított függvényei

összegzés

- Uf: $s = \text{SZUMMA}(i=1..n, x[i])$

- Példa:

- Uf: $s = \text{SZUMMA}(i=1..n, \text{szamok}[i])$

- Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
  
int s = Mintak.Szumma(szamok);
```

```
int s = szamok.Sum();
```

- Példa (transzformátorfüggvény):

- Uf: $s = \text{SZUMMA}(i=1..n, \text{diakok}[i].\text{jegy})$

- Kód:

```
int s = Mintak.Szumma(diakok, diak => diak.jegy);
```

```
int s = diakok.Sum(diak => diak.jegy);
```


Tételek általánosított függvényei

általános összegzés

- Uf: $s = \text{SZUMMA}(i=e..u, x[i], \text{kezd}, \text{add})$
- Példa:
 - Uf: $p = \text{SZUMMA}(i=1..n, \text{szamok}[i], 1, \text{add})$
 $\text{add}(s, p) = s * p$
 - Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
  
int s = Mintak.Szumma(szamok, 1, (s, p) => s * p);  
  
int s = szamok.Aggregate(1, (s, p) => s * p);
```

Tételek általánosított függvényei

megszámolás

- Uf: $db = \text{DARAB}(i=e..u, T(x[i]))$
- Példa:
 - Uf: $db = \text{SZUMMA}(i=1..n, \text{szamok}[i] < 0)$
 - Kód:

```
int[] szamok = { 2, 4, -7, 5, 3, -2, 1 };  
int n = szamok.Length;  
  
int db = Mintak.Darab(szamok, e => e < 0);
```

```
int db = szamok.Count(e => e < 0);
```

Tételek általánosított függvényei

maximumkiválasztás

- Uf: $(\text{maxind}, \text{maxért}) = \text{MAX}(i=1..n, x[i])$
- Példa: $(\text{maxi}, \text{maxé}) = \text{MAX}(i=1..n, \text{szamok}[i])$

- Kód:

```
int[] szamok = { 2, 4, 7, 5, 3, 2, 1 };  
int n = szamok.Length;  
(int maxi, int maxe) = Mintak.Max(szamok);
```

```
int maxe = szamok.Max();
```

- Példa (transzformátorfüggvény):

- Uf: $(\text{ind}, \text{diák}) = \text{MAX}(i=1..n, \text{diákok}[i].\text{jegy})$

- Kód:

```
struct Diak { public string nev; public int jegy; }  
(int ind, int diak) = Mintak.Max(diakok, diak => diak.jegy);
```

```
int maxe = diakok.Max(diak => diak.jegy);
```

- Példa (összehasonlító függvény):

- Uf: $(\text{ind}, \text{diák}) = \text{MAX}(i=1..n, \text{diákok}[i].\text{jegy})$

- Kód:

```
(int ind, int diak) = Mintak.Max(diakok,  
                                (d1, d2) => d1.jegy > d2.jegy);
```

Tételek általánosított függvényei

feltételes maximumkeresés

- Uf: $(\text{van}, \text{maxind}, \text{maxért}) = \text{FELTMAX}(i=1..n, x[i], T(x[i]))$
- Példa: $(\text{van}, \text{maxi}, \text{maxé}) = \text{FELTMAX}(i=1..n, \text{szamok}[i], \text{szamok}[i] < 0)$

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };
int n = szamok.Length;
(bool van, int maxi, int maxe) = Mintak.FeltMax(szamok, e => e < 0);
```

```
int maxe = szamok.Where(e => e < 0).Max();
```

- Példa (transzformátorfüggvény):

- Uf: $(\text{van}, \text{maxi}, \text{maxé}) = \text{FELTMAX}(i=1..n, \text{diakok}[i].\text{jegy}, \text{diakok}[i].\text{jegy} < 5)$

- Kód:

```
struct Diak { public string nev; public int jegy; }
(bool van, int ind, int diak) = Mintak.FeltMax(diakok,
    diak => diak.jegy, diak => diak.jegy < 5);
```

```
int diak = diakok.Where(dk => dk.jegy < 5).Max(dk => dk.jegy);
```

- Példa (összehasonlító függvény):

- Uf: $(\text{van}, \text{maxi}, \text{maxé}) = \text{FELTMAX}(i=1..n, \text{diakok}[i].\text{jegy}, \text{diakok}[i].\text{jegy} < 5)$

```
(bool van, int ind, int diak) = Mintak.FeltMax(diakok,
    diak => diak.jegy < 5, (d1, d2) => d1.jegy > d2.jegy);
```



Tételek általánosított függvényei

keresés

- Uf: $(\text{van}, \text{ind}) = \text{KERES}(\text{i}=1..n, T(x[\text{i}]))$
- Példa:
 - Uf: $(\text{van}, \text{ind}) = \text{KERES}(\text{i}=1..n, \text{szamok}[\text{i}] < 0)$
 - Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;
```

```
(bool van, int ind) = Mintak.Keres(szamok, e => e < 0);
```

```
int ertek = szamok.FirstOrDefault(e => e < 0, 0);
```

Tételek általánosított függvényei

eldöntés

- Uf: `van=VAN(i=1..n, T(x[i]))`

- Példa:

- Uf: `van=VAN(i=1..n, szamok[i]<0)`

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
bool van = Mintak.Van(szamok, e => e < 0);
```

```
bool van = szamok.Any(e => e < 0);
```

- Példa:

- Uf: `mind=MIND(i=1..n, szamok[i]>0)`

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
bool mind = Mintak.Mind(szamok, e => e > 0);
```

```
bool van = szamok.All(e => e > 0);
```

Tételek általánosított függvényei

kiválasztás

- Uf: $\text{ind} = \text{KIVÁLASZT}(i \geq e, T(x[i]))$

- Példa:

- Uf: $\text{ind} = \text{KIVÁLASZT}(i \geq 1, \text{szamok}[i] < 0)$

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
int ind = Mintak.Kivalaszt(szamok, e => e < 0);
```

```
int ertek = szamok.First(e => e < 0);
```

Tételek általánosított függvényei

másolás

- Uf: $y = \text{MÁSOL}(i=1..n, f(x[i]))$

- Példa:

- Uf: $y = \text{MÁSOL}(i=1..n, \text{abs}(\text{szamok}[i]))$

- Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;
```

```
int[] y = Mintak.Masol(szamok, e => Math.Abs(e));
```

```
int[] y = szamok.Select(e => Math.Abs(e)).ToArray();
```

- Példa (értékek és indexek):

```
int[] y = diakov  
    .Select((e, i) => (ertekek: e, ind: i))  
    .Select(e => e.ind)  
    .ToArray();
```


Tételek általánosított függvényei

kiválogatás

- Uf: $(db, y) = \text{KIVÁLOGAT}(i=1..n, T(x[i]), x[i])$
- Példa:
 - Uf: $(db, y) = \text{KIVÁLOGAT}(i=1..n, \text{szamok}[i] < 0, \text{szamok}[i])$
 - Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
int[] y = Mintak.Kivalogat(szamok, e => e < 0);
```
- Példa:
 - Uf: $(db, y) = \text{KIVÁLOGAT}(i=1..n, \text{szamok}[i] < 0, i)$
 - Kód:

```
int[] y = Mintak.Kivalogat(szamok, e => e < 0, (e, i) => i);  
  
int[] y = szamok  
    .Select((e,i) => (ertek: e, ind: i))  
    .Where(e => e.ertek < 0)  
    .Select(e => e.ind).ToArray();
```

Példa

legjobb jó tanuló

	1	2	3	m=4
1	5	5	5	3
2	5	4	4	4
3	5	3	2	4
4	5	4	5	5
n=5	5	5	4	4



Feladat:

Egy egész számokat tartalmazó mátrixban **melyik az a sor**,
amiben **csak 4-es és 5-ös van**, és az **összege a legnagyobb?**

Specifikáció:

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

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

Fv: összeg: $\mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: jó: $\mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n] : (\forall \text{oszlop} \in [1..m] : (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb},) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

	1	2	3	m=4	
1	5	5	5	3	
2	5	4	4	4	17
3	5	3	2	4	→ van=igaz → legjobb=4
4	5	4	5	5	19
n=5	5	5	4	4	18

Specifikáció:

Be: $n \in \mathbb{N}$, $m \in \mathbb{N}$, $\text{jegyek} \in \mathbb{N}[1..n, 1..m]$

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

Fv: $\text{összeg}: \mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: $\text{jó}: \mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n]: (\forall \text{oszlop} \in [1..m]: (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb}) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

```
static void Main(string[] args) {
    int n; int m; int[][] jegyek;
    bool van; int legjobb;

    (n, m, jegyek) = beolvas();
    (van, legjobb, _) = Mintak.Max(jegyek,
        diak => Mintak.Szumma(diak),
        diak => Mintak.Mind(diak, jegy => 4 <= jegy && jegy <= 5)
    );
    kiir(van, legjobb);
}

static (int n, int m, int[][] jegyek) beolvas() {
    int n, m;
    int[][] jegyek;

    Console.Write("Varazstanoncok szama = ");
    int.TryParse(Console.ReadLine(), out n);
    Console.Write("Jegyek szama = ");
    int.TryParse(Console.ReadLine(), out m);
    jegyek = new int[n][];
    for (int i = 1; i <= n; i++) {
        jegyek[i - 1] = new int[m];
        for (int j = 1; j <= m; j++) {
            Console.Write("{0}. varazstanonc {1} jegye = ", i, j);
            int.TryParse(Console.ReadLine(), out jegyek[i - 1][j - 1]);
        }
    }
    return (n, m, jegyek);
}
```

Specifikáció:

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

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

Fv: összeg: $\mathbb{N} \rightarrow \mathbb{N}$,

$\text{összeg}(\text{diák}) = \text{SZUMMA}(\text{tantárgy} = 1..m, \text{jegyek}[\text{diák}, \text{tantárgy}])$

Fv: jó: $\mathbb{N} \rightarrow \mathbb{L}$,

$\text{jó}(\text{diák}) = \text{MIND}(\text{tantárgy} = 1..m, 4 \leq \text{jegyek}[\text{diák}, \text{tantárgy}] \leq 5)$

Ef: $\forall \text{sor} \in [1..n]: (\forall \text{oszlop} \in [1..m]: (1 \leq \text{jegyek}[\text{sor}, \text{oszlop}] \leq 5))$

Uf: $(\text{van}, \text{legjobb}) = \text{FELTMAX}(\text{diák} = 1..n, \text{összeg}(\text{diák}), \text{jó}(\text{diák}))$

```
static void Main(string[] args) {
    int n; int m; int[][] jegyek;
    bool van; int legjobb;

    (n, m, jegyek) = beolvas();
    (van, legjobb, _) = Mintak.Max(jegyek,
        diak => diak.Sum(),
        diak => diak.All(jegy => 4 <= jegy && jegy <= 5)
    );
    kiir(van, legjobb);
}

static (int n, int m, int[][] jegyek) beolvas() {
    int n, m;
    int[][] jegyek;

    Console.Write("Varazstanoncok szama = ");
    int.TryParse(Console.ReadLine(), out n);
    Console.Write("Jegyek szama = ");
    int.TryParse(Console.ReadLine(), out m);
    jegyek = new int[n][];
    for (int i = 1; i <= n; i++) {
        jegyek[i - 1] = new int[m];
        for (int j = 1; j <= m; j++) {
            Console.Write("{0}. varazstanonc {1} jegye = ", i, j);
            int.TryParse(Console.ReadLine(), out jegyek[i - 1][j - 1]);
        }
    }
    return (n, m, jegyek);
}
```

Összefoglalás



Minták általánosított függvényei

- Uf: `van=VAN(i=e..u, T(i))`
- Példa:
 - Uf: `van=VAN(i=1..n, szamok[i]<0)`
 - Kód:

```
int[] szamok = { -2, 4, 7, -5, 3, 2, -1 };  
int n = szamok.Length;  
  
bool van = Mintak.Van(1, n, i => szamok[i - 1] < 0);  
  
bool van = Mintak.Van(szamok, e => e < 0);  
  
bool van = szamok.Any(e => e < 0);
```