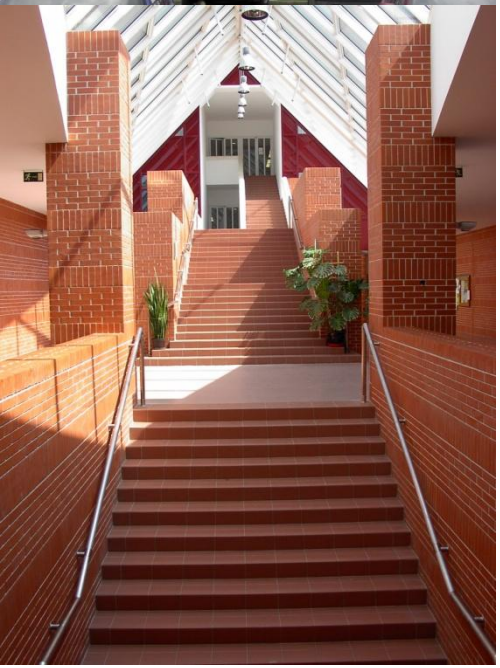
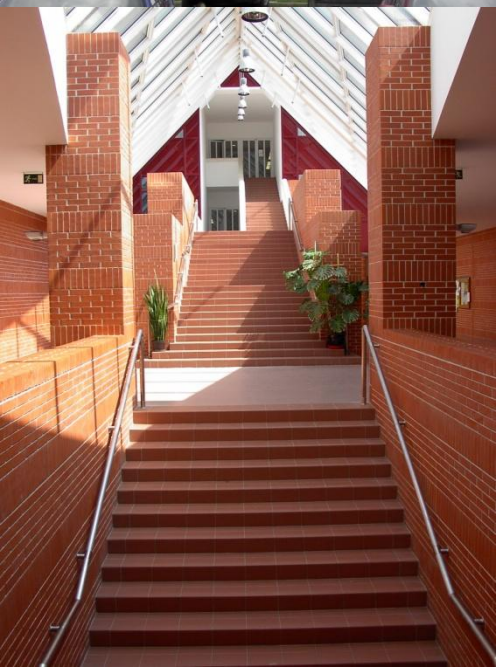
Programozási alapismeretek
8. előadás



Tartalom

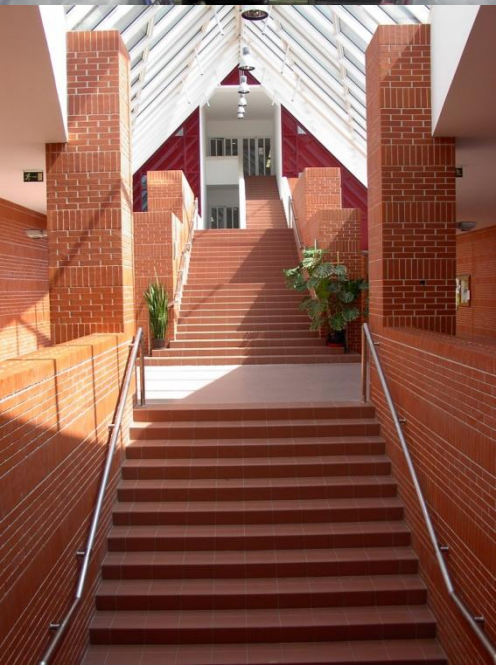
- Programozási tételek alkalmazása
- Sorozatszámítás –
  rekordok: jövedelem
- Maximum-kiválasztás –
  rekordok: legkorábbi születésnap
- Függvény a feltételben
- Mátrixok
- Rekordok vektora
- Vektorok rekordja



Tartalom

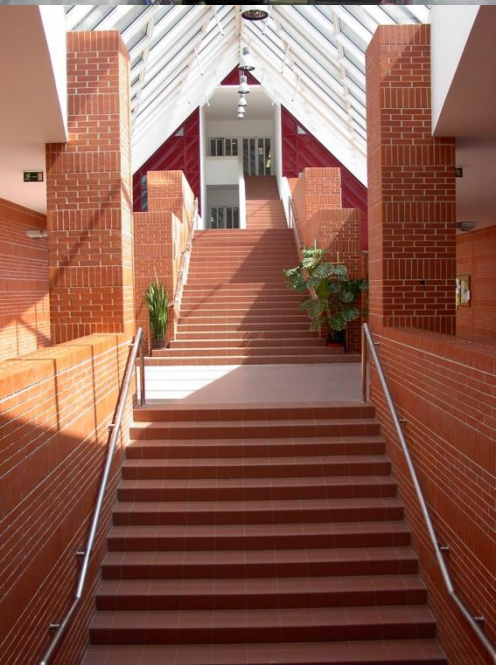
- Programozási tételek alkalmazása
- Sorozatszámítás –
  rekordok: jövedelem
- Maximum-kiválasztás –
  rekordok: legkorábbi születésnap
- Függvény a feltételben
- Mátrixok
- Rekordok vektora
- Vektorok rekordja

A nagy sóhajtások előadása



Tartalom

- Programozási tételek alkalmazása
- Sorozatszámítás –
  rekordok: jövedelem
- Maximum-kiválasztás –
  rekordok: legkorábbi születésnap
- Függvény a feltételben
- Mátrixok
- Rekordok vektora
- Vektorok rekordja



Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. Elsőfokú árvízvédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, másodfokút, ha meghaladja a 900 centimétert és harmadfokút, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

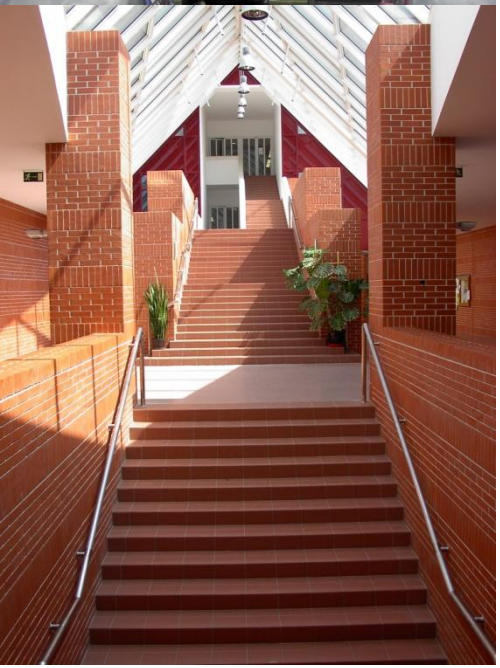
Adjuk meg, hogy hány folyószakaszon volt árvíz!

Programozási tételek alkalmazása

Feladat:

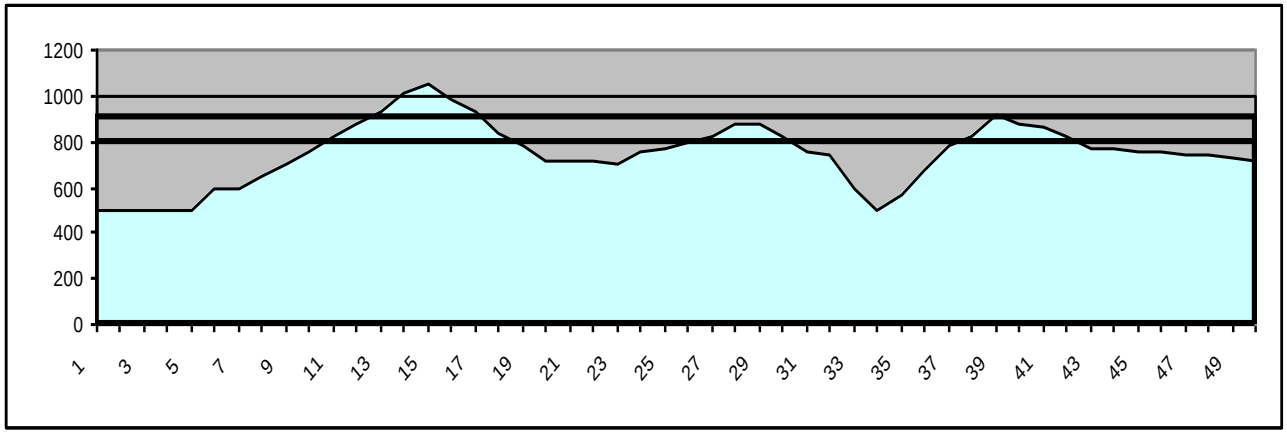
Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!



Adjuk meg, hogy **hány** folyószakaszon volt árvíz!

A feladat szöveg alapján ez egy **megszámolás** programozási tétel.

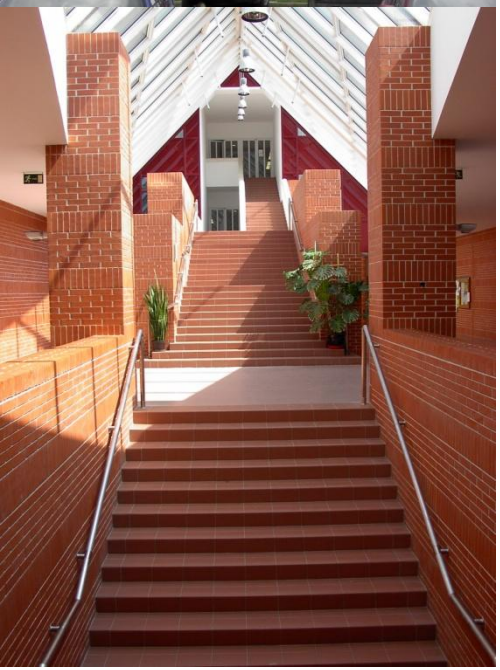


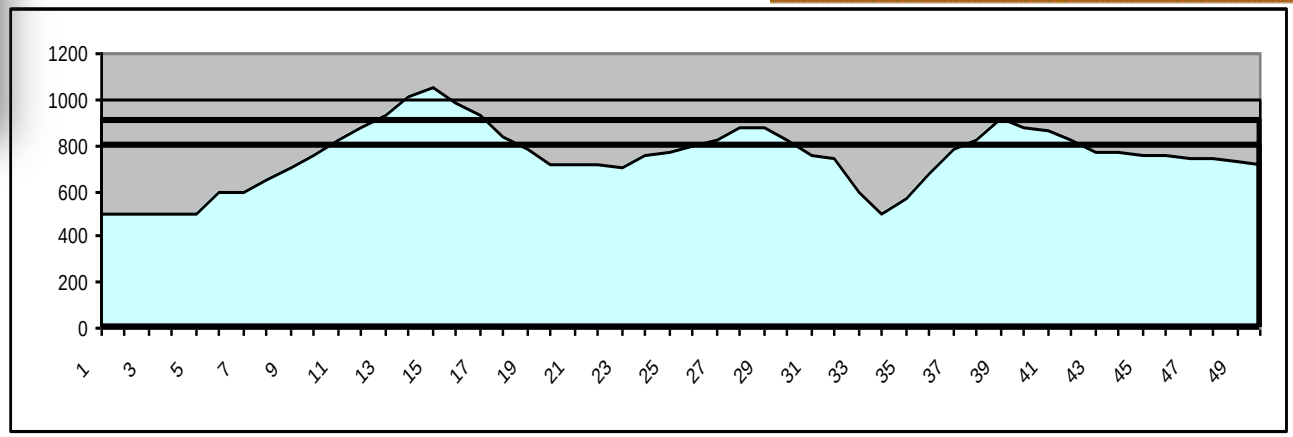
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_1$:

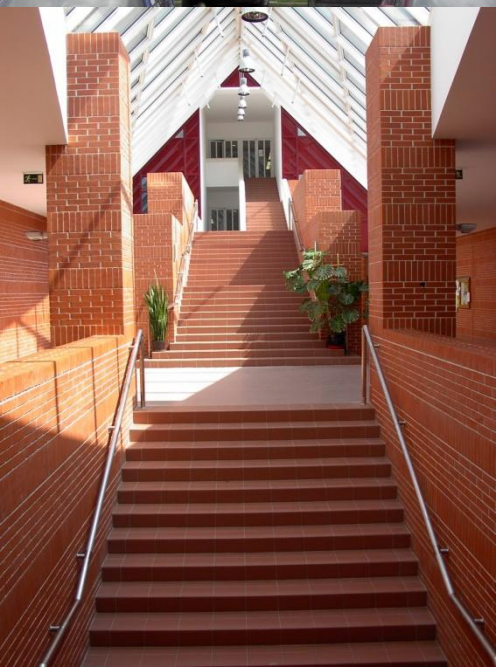
- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$

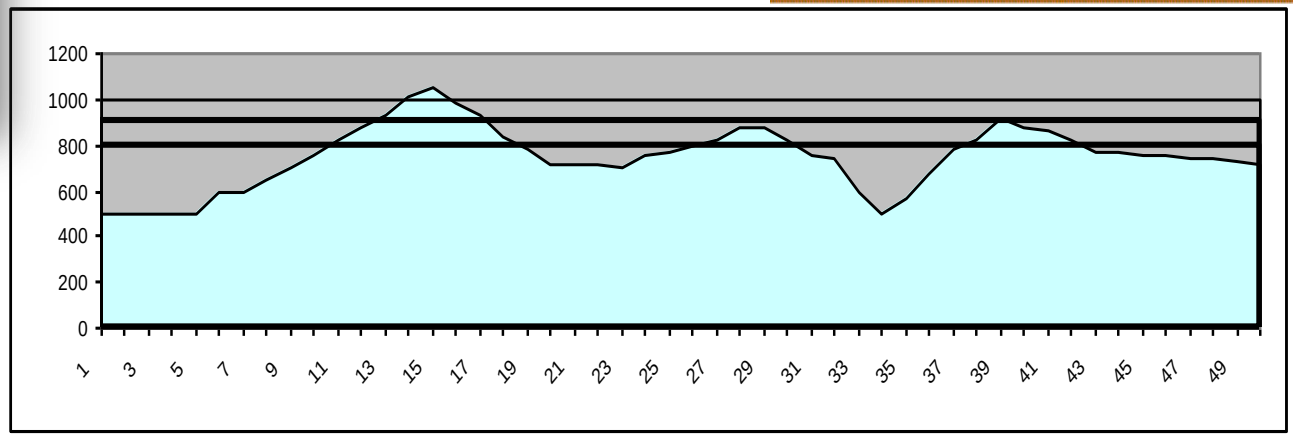
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_1$:

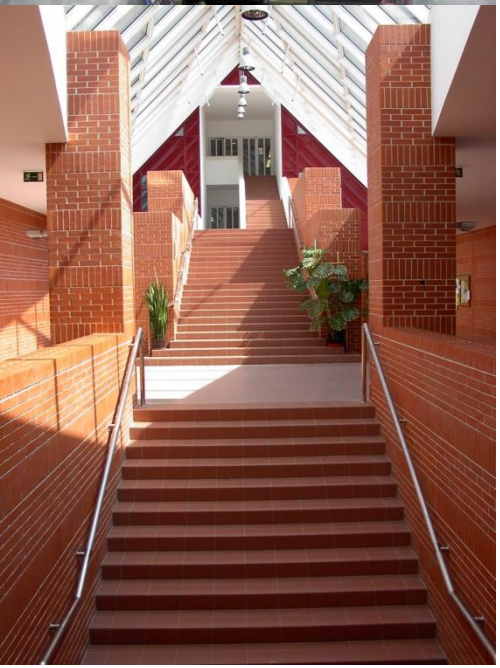
- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$

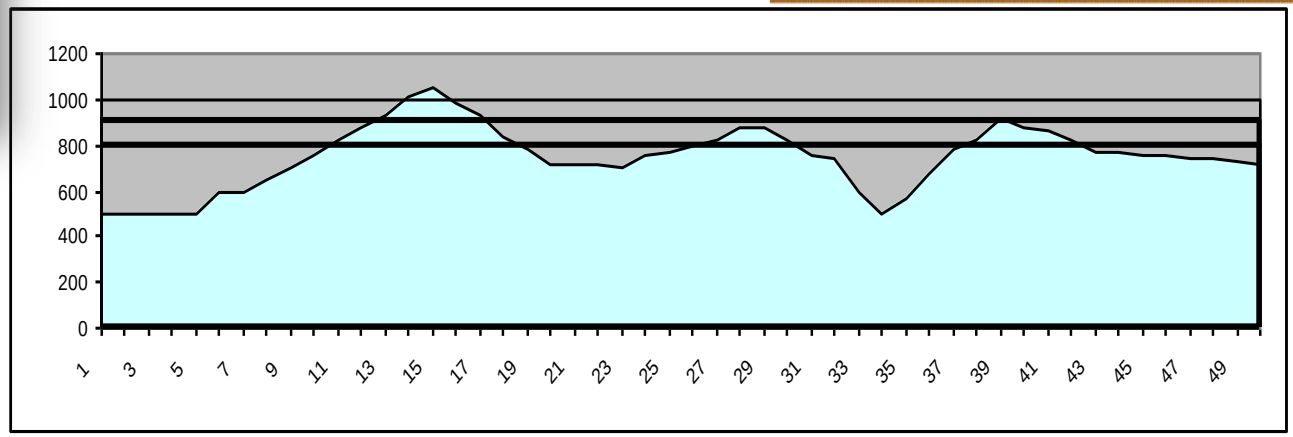
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_1$:

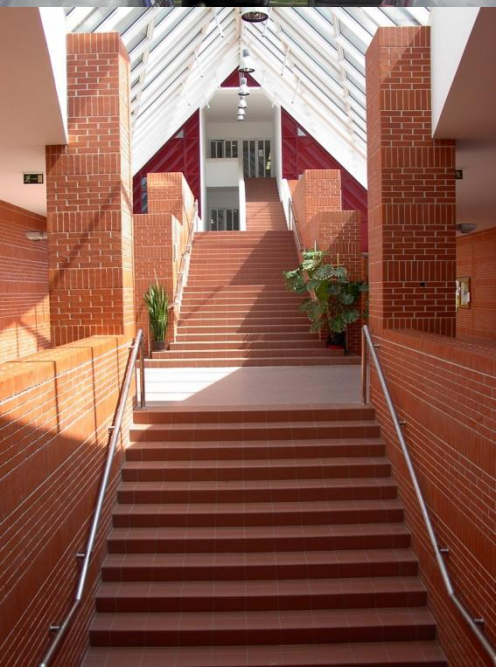
- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –

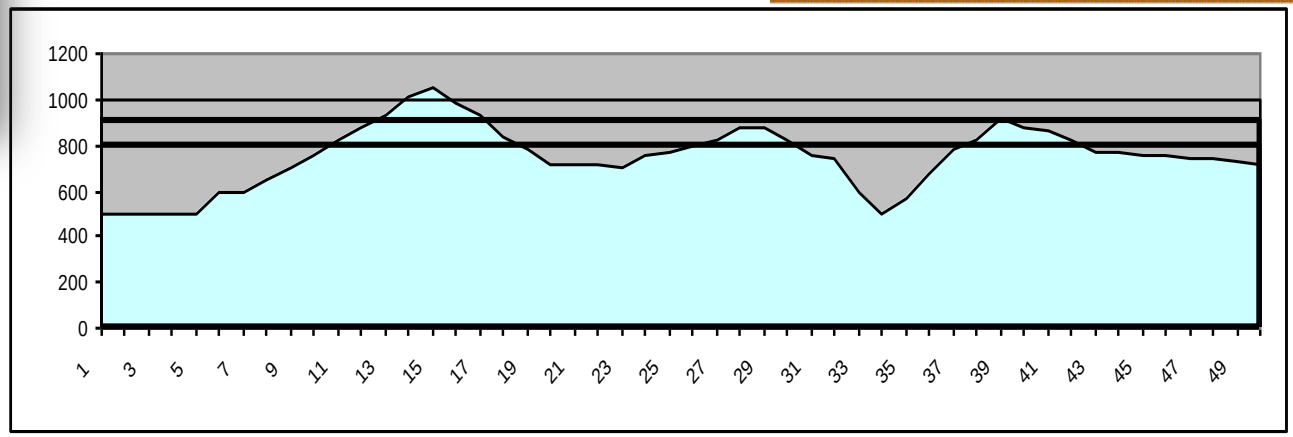
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_1$:

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $Db = \sum_{i=1}^{N-1} 1$

$F_i \leq 800$ és $F_{i+1} > 800$ vagy $i=1$ és $F_i > 800$

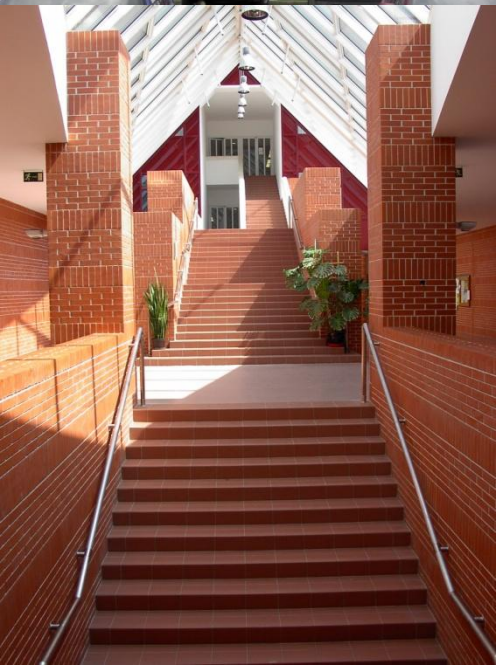
Azaz annyi szakaszon volt árvíz, ahány helyen árvíz kezdődött vagy már az elején is árvíz volt.

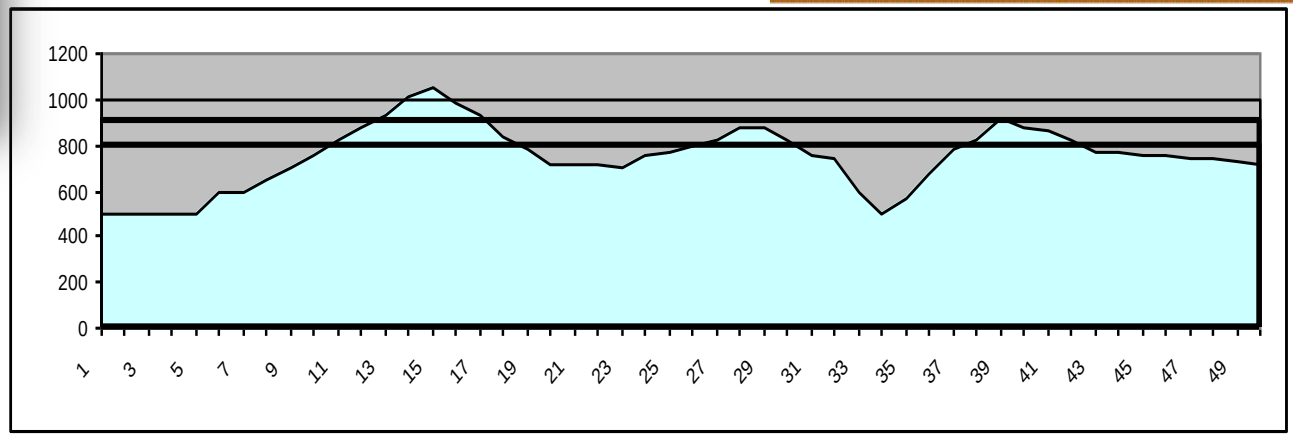
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_2$:

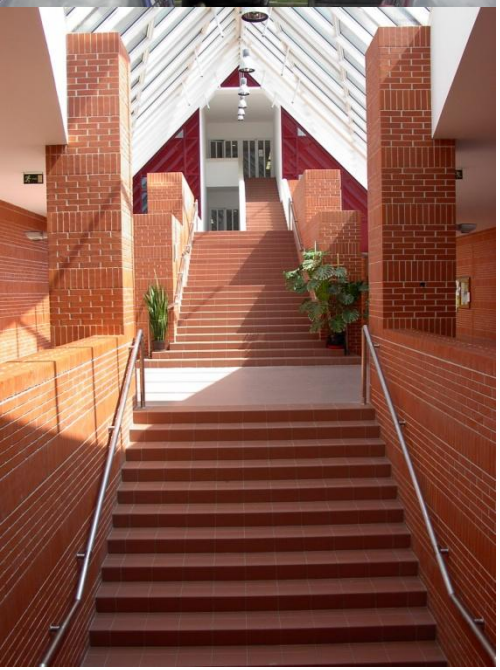
- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –

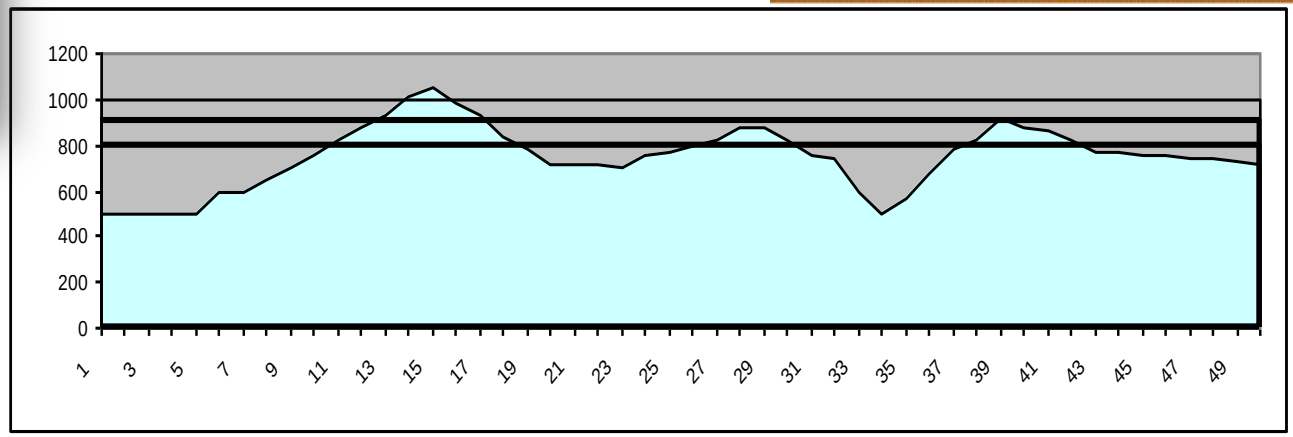
Programozási tételek alkalmazása

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. **Elsőfokú** árvívédelmi készültséget kell elrendelni, ha a magasság meghaladja a 800 centimétert, **másodfokút**, ha meghaladja a 900 centimétert és **harmadfokút**, ha meghaladja az 10 métert. Folyószakasznak nevezzük a leghosszabb egymás mellett levő egyforma tulajdonságú mérésekből álló sorozatokat. **Árvíz**nek nevezzük azt a folyószakaszt, ahol minden hely legalább elsőfokú készültségű.

Adjuk meg, hogy hány folyószakaszon volt árvíz!





Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $Db = \sum_{i=2}^{N} 1$

$F_i \leq 800$ és $F_{i-1} > 800$ vagy $i=N$ és $F_i > 800$

Azaz annyi szakaszon volt árvíz, ahány helyen árvíz végződött vagy már a végén is árvíz volt.

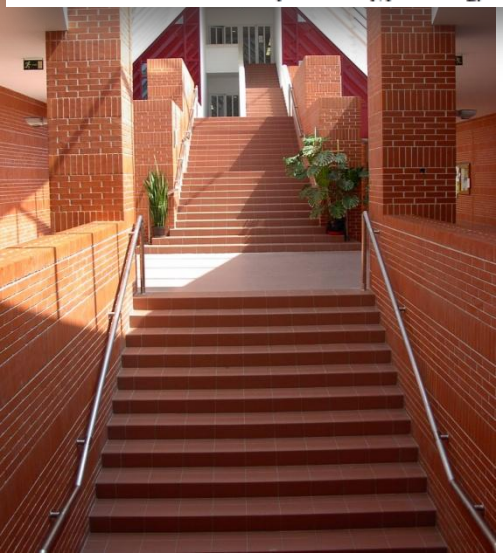
Programozási tételek alkalmazása

Algoritmus$_1$:

Specifikáció$_1$:

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $Db = \sum_{i=1}^{N-1} 1$

$F_i \leq 800$ és $F_{i+1} > 800$ vagy $i=1$ és $F_i > 800$



Változó
i:Egész

I F[1]>800 N	
Db:=1	Db:=0

i=1..N–1

I F[i]≤800 és F[i+1]>800 N	
Db:=Db+1	—

Különválasztva a csak egyszer (i=1-nél) teljesülőt a többitől. (Optimizálás.)

Programozási tételek alkalmazása

Algoritmus$_2$:

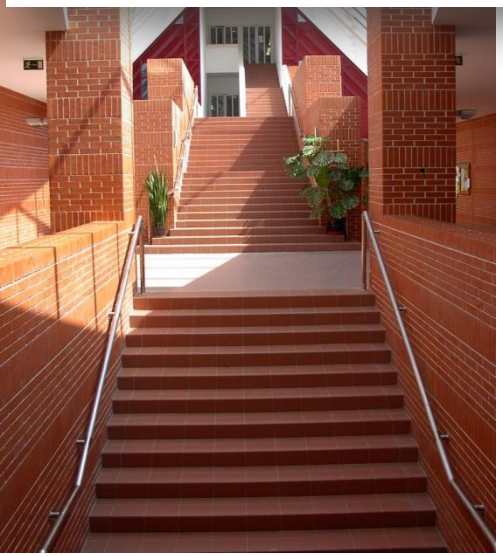
Változó
i:Egész

I	F[N]>800	N
Db:=1		Db:=0
i=2..N		
I	F[i]≤800 és F[i-1]>800	N
Db:=Db+1		—

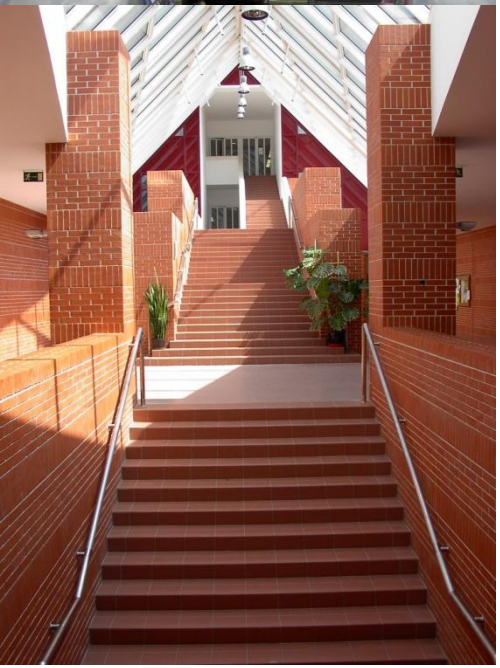
Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $Db = \sum_{i=2}^{N} 1$

$F_i \leq 800$ és $F_{i-1} > 800$ vagy $i=N$ és $F_i > 800$



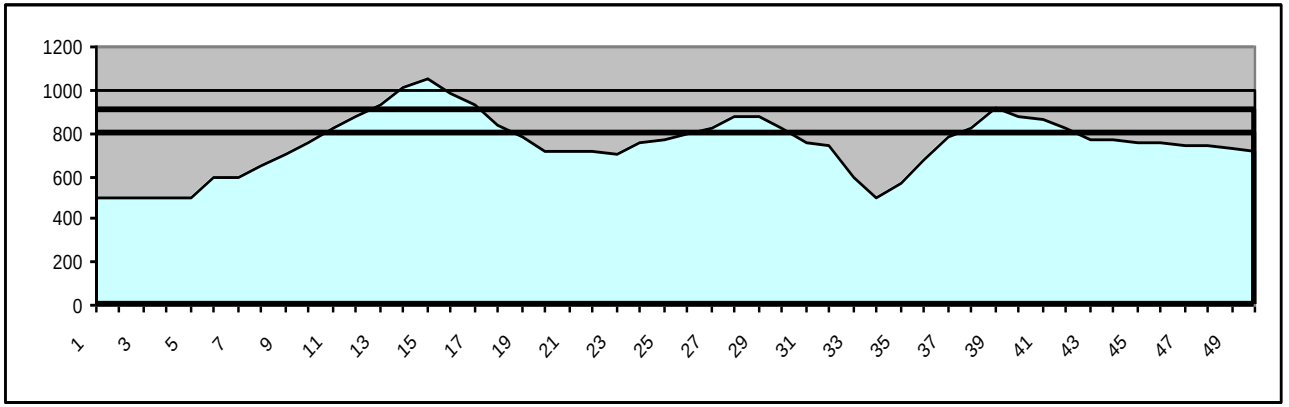
Különválasztva a csak egyszer (i=N-nél) teljesülőt a többitől. (Optimizálás.)



Programozási tételek alkalmazása

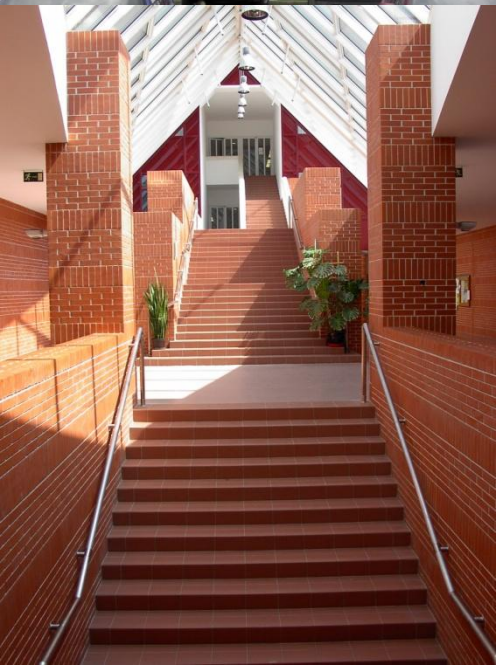
Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



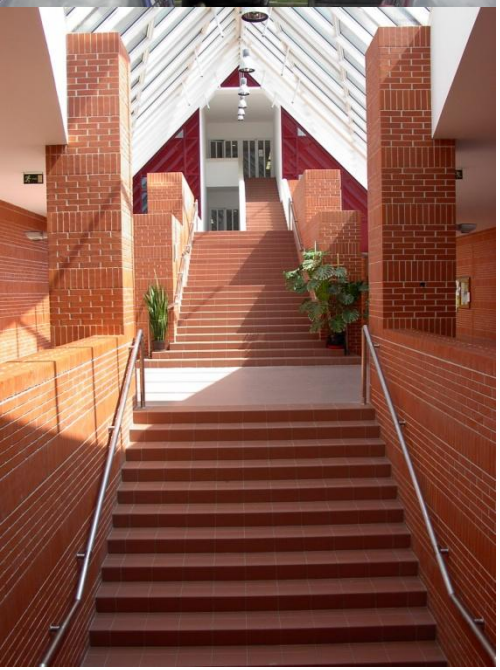
Programozási tételek alkalmazása

Specifikáció: (kiválogatás+másolás)

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



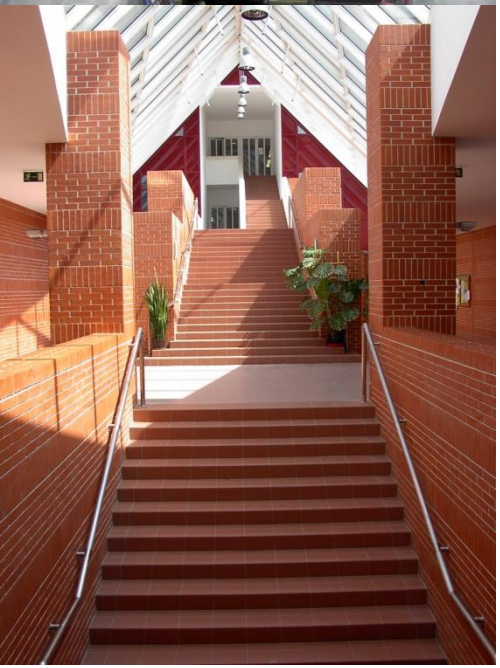
Programozási tételek alkalmazása

Specifikáció: (kiválogatás+másolás)

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



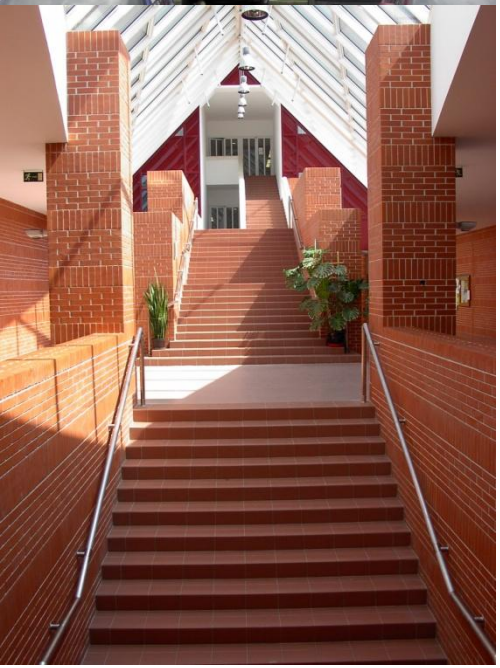
Programozási tételek alkalmazása

Specifikáció: (kiválogatás+másolás)

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$
- Előfeltétel: $N \geq 1$ és $F_1 \leq 800$ és $F_N \leq 800$

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



Programozási tételek alkalmazása

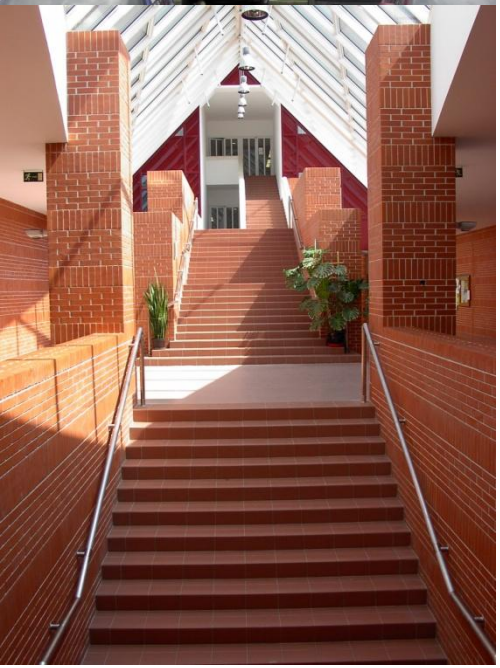
Specifikáció: (kiválogatás+másolás)

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$
- Előfeltétel: $N \geq 1$ és $F_1 \leq 800$ és $F_N \leq 800$
- Utófeltétel: $Db = \sum_{\substack{i=1 \\ F_i \leq 800 \text{ és } F_{i+1} > 800}}^{N-1} 1$ és

$$\forall i\ (1 \leq i \leq Db): H_i = \text{árvízvég}_i - \text{árvízkezdet}_i - 1$$

Feladat:

Egy folyón N helyen mérik a vízállást, amit egy referenciamagassághoz képest centiméterben adnak meg. … Adjuk meg az árvizek hosszát! Feltehetjük, hogy az első és az utolsó mérésnél nem volt árvíz.



Programozási tételek alkalmazása

Specifikáció: (kiválogatás+másolás)

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$
- Előfeltétel: $N \geq 1$ és $F_1 \leq 800$ és $F_N \leq 800$
- Utófeltétel: $Db = \sum_{\substack{i=1 \\ F_i \leq 800 \text{ és } F_{i+1} > 800}}^{N-1} 1$ és

$$\forall i\ (1 \leq i \leq Db): H_i = \text{árvízvég}_i - \text{árvízkezdet}_i - 1$$

- Definíció:

$\text{árvízvég} \in \mathbb{N}^{Db}$ …

$\text{árvízkezdet} \in \mathbb{N}^{Db}$ …

Programozási tételek alkalmazása

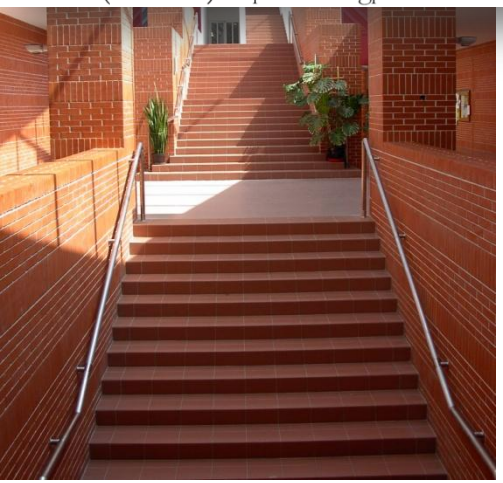
Algoritmus: (kiválogatás⊃megszámolás)

Egyszerűsítő ötlet: a megfelelő számlálót árvíz-kezdetnél nullázzuk, árvíznél pedig növeljük.

Specifikáció:

- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$
- Előfeltétel: $N \geq 1$ és $F_1 \leq 800$ és $F_N \leq 800$
- Utófeltétel: $Db = \sum_{\substack{i=1 \\ F_i \leq 800 \text{ és } F_{i-1} > 800}}^{N-1} 1$ és

$\forall i\ (1 \leq i \leq Db)$: $H_i = \text{árvízvég}_i - \text{árvízkezdet}_i - 1$



Programozási tételek alkalmazása

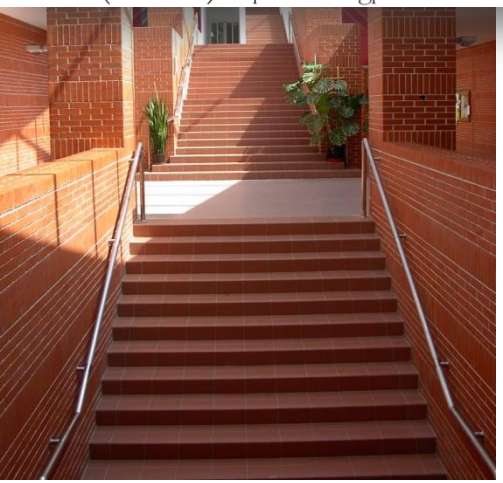
Algoritmus: (kiválogatás⊃megszámolás)

Egyszerűsítő ötlet: a megfelelő számlálót árvízkezdetnél nullázzuk, árvíznél pedig növeljük.

Specifikáció:

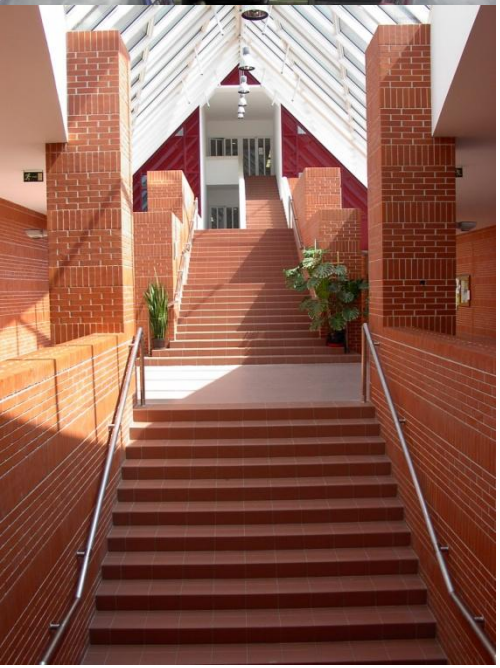
- Bemenet: $N \in \mathbb{N}, F \in \mathbb{N}^N$
- Kimenet: $Db \in \mathbb{N}, H \in \mathbb{N}^{Db}$
- Előfeltétel: $N \geq 1$ és $F_1 \leq 800$ és $F_N \leq 800$
- Utófeltétel: $Db = \sum_{i=1}^{N-1} 1$ és $F_i \leq 800$ és $F_{i+1} > 800$

$\forall i\ (1 \leq i \leq Db)$: H_i=árvízvég$_i$–árvízkezdet$_i$–1



Változó
i:Egész

Db:=0	
i=1..N–1	
I: F[i]≤800 és F[i+1]>800 : N	
Db:=Db+1	—
H[Db]:=0	
I: F[i]>800 : N	
H[Db]:=H[Db]+1	—

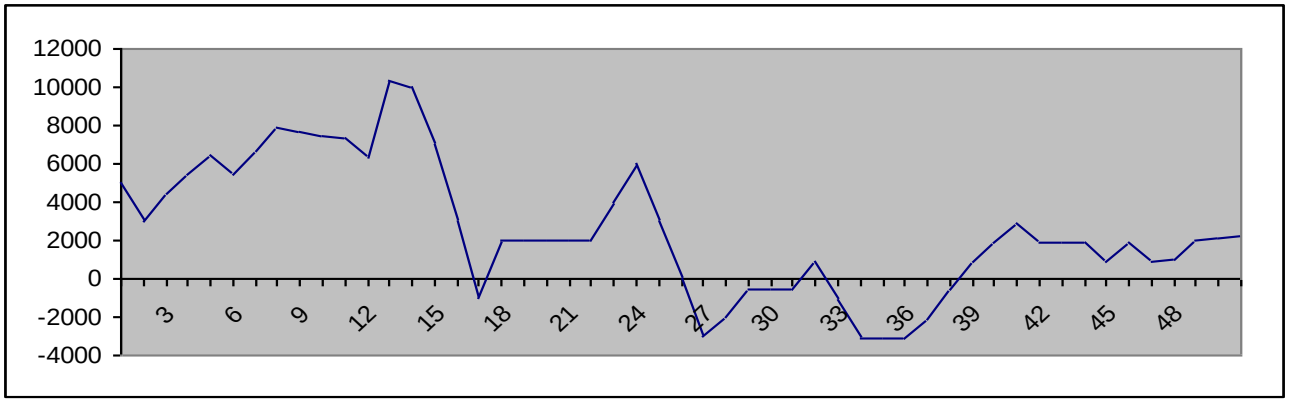


Programozási tételek alkalmazása

Feladat:

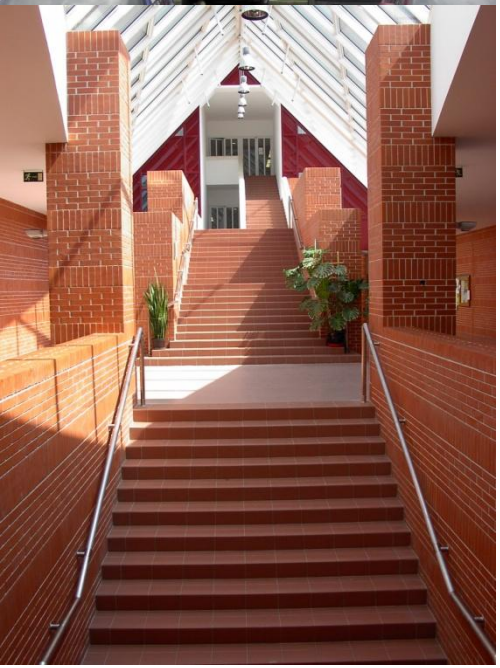
Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor **mely** napon kellett először kölcsön kérnünk?

Összegzés és **keresés** programozási tétel!



Feladat:

Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor mely napon kellett először kölcsön kérnünk?



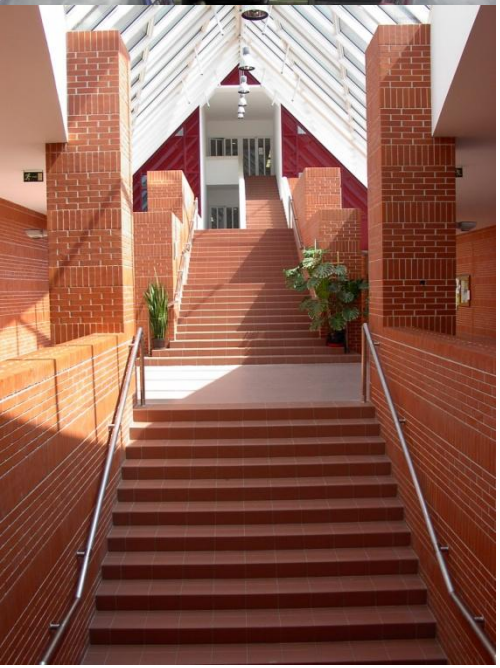
Programozási tételek alkalmazása

Specifikáció: (keresés+összegzés)

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$

Feladat:

Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor mely napon kellett először kölcsön kérnünk?



Programozási tételek alkalmazása

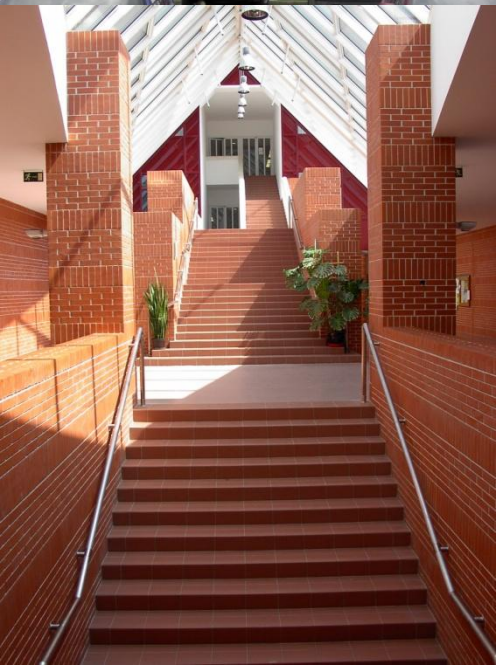
Specifikáció: (keresés+összegzés)

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$
- Kimenet: $Van \in \mathbb{L}, Nap \in \mathbb{N}$

Feladat:

Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor mely napon kellett először kölcsön kérnünk?

Programozási tételek alkalmazása

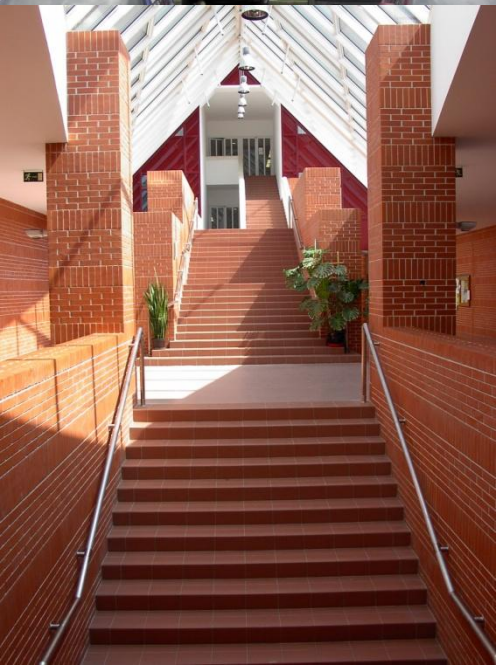


Specifikáció: (keresés+összegzés)

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$
- Kimenet: $Van \in \mathbb{L}, Nap \in \mathbb{N}$
- Előfeltétel: $X>0$

Feladat:

Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor mely napon kellett először kölcsön kérnünk?



Programozási tételek alkalmazása

Specifikáció: (keresés+összegzés)

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$
- Kimenet: $Van \in \mathbb{L}, Nap \in \mathbb{N}$
- Előfeltétel: $X>0$
- Utófeltétel: $Van = \exists i (1 \le i \le N): \quad X + \sum_{j=1}^{i} P_j \le 0 \quad$ és

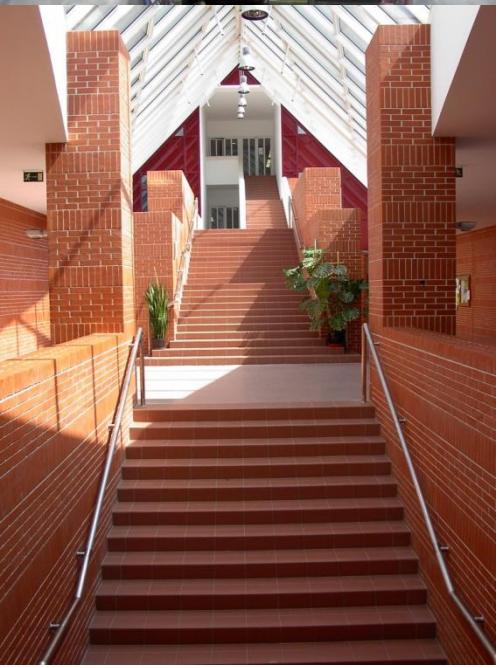
$$Van \rightarrow$$

$$\left(1 \le Nap \le N \quad \text{és} \quad X + \sum_{j=1}^{Nap} P_j \le 0 \quad \text{és} \right.$$

$$\left. \forall i (1 \le i < Nap) \quad \text{és} \quad X + \sum_{j=1}^{i} P_j > 0 \right)$$

Feladat:

Lóversenyre járunk. N napon át naponta feljegyeztük, hogy mennyit nyertünk (≥0) vagy vesztettünk (<0). Ha kezdetben X forintunk volt, akkor mely napon kellett először kölcsön kérnünk?



Programozási tételek alkalmazása

Specifikáció: (keresés+összegzés)

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$
- Kimenet: $Van \in \mathbb{L}, Nap \in \mathbb{N}$
- Előfeltétel: $X>0$
- Utófeltétel: $Van = \exists i (1 \le i \le N): \quad X + \sum_{j=1}^{i} P_j \le 0 \quad$ és

$$Van \rightarrow$$

$$\left(1 \le Nap \le N \quad \text{és} \quad X + \sum_{j=1}^{Nap} P_j \le 0 \quad \text{és} \right.$$

$$\left. \forall i (1 \le i < Nap) \quad \text{és} \quad X + \sum_{j=1}^{i} P_j > 0 \right)$$

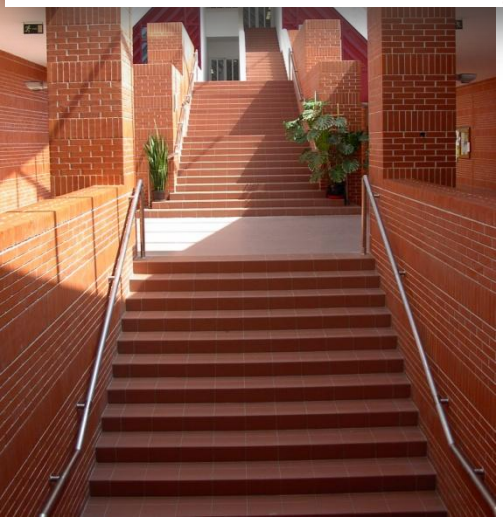
Rövidebben: $(Van, Nap) = Keres_{i=1}^{N}{}_{X+\sum_{j=1}^{i} P_j \le 0}$

Programozási tételek alkalmazása

Specifikáció:

- Bemenet: $N,X \in \mathbb{N}, P \in \mathbb{Z}^N$
- Kimenet: $Van \in \mathbb{L}$, $Nap \in \mathbb{N}$
- Előfeltétel: $X>0$
- Utófeltétel: $Van=\exists i(1 \le i \le N): X+\sum_{j=1}^{i} P_j \le 0$ és $Van \rightarrow$

$$(1 \le Nap \le N \text{ és } X+\sum_{j=1}^{Nap} P_j \le 0 \text{ és } \forall i(1 \le i < Nap) \text{ és } X+\sum_{j=1}^{i} P_j > 0)$$



Algoritmus: (keresés+összegzés)

Változó
i,S:Egész

S:=X

i:=1

i≤N és S+P[i]>0

- S:=S+P[i]
- i:=i+1

Van:=i≤N

I Van N	
Nap:=i	—

Sorozatszámítás

rekordok: jövedelem

Feladat:

Mennyi a jövedelmünk, ha ismertek a bevételeink és a vele szemben elszámolt kiadásaink?

Így specifikáltuk ezt a 4. előadásban.

Specifikáció:

Bemenet: $N \in \mathbb{N}$,
$Be, Ki \in \mathbb{Z}^N$

Kimenet: $S \in \mathbb{Z}$

Előfeltétel: $\forall i\ (1 \le i \le N)$: $Be_i, Ki_i \ge 0$

Utófeltétel: $S = \sum_{i=1}^{N} Be_i - Ki_i$

Sorozatszámítás

rekordok: jövedelem

Így specifikálhatjuk a 6. előadás után.

Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}$,

$Jöv \in Nyer^N$, $Nyer = Be \times Ki$, $Be, Ki = \mathbb{N}$

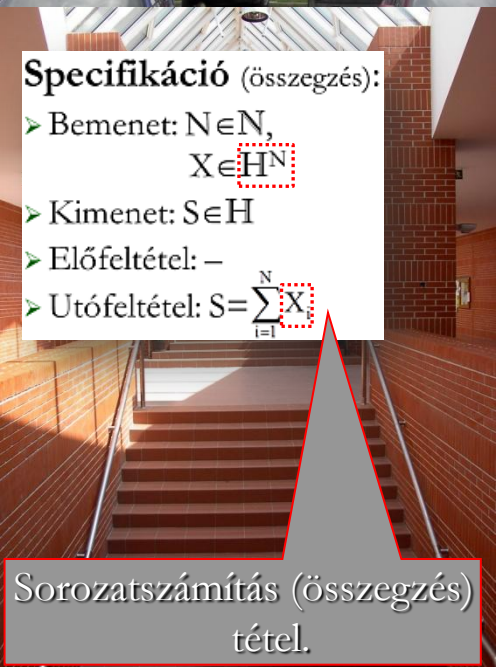
Sorozatszámítás

rekordok: jövedelem

Így specifikálhatjuk a 6. előadás után.

Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}$,
  $Jöv \in Nyer^N$, $Nyer = Be \times Ki$, $Be, Ki = \mathbb{N}$
- Kimenet: $S \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $S = \sum_{i=1}^{N} Jöv_i.be - Jöv_i.ki$



Sorozatszámítás

rekordok: jövedelem

Így specifikálhatjuk a 6. előadás után.

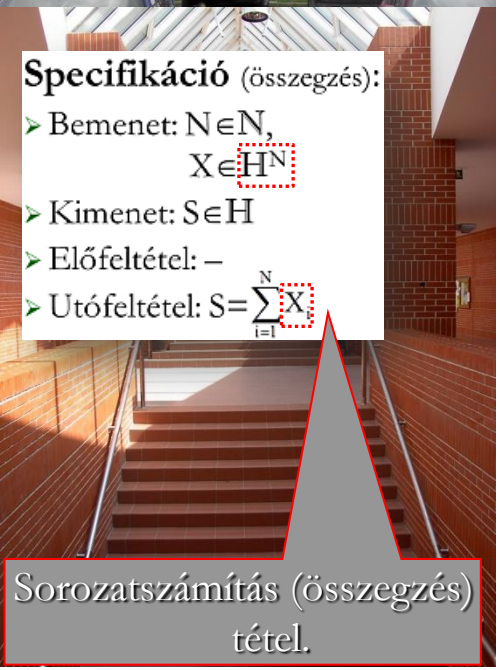
Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}$,
  $Jöv \in Nyer^N$, $Nyer = Be \times Ki$, $Be, Ki = \mathbb{N}$
- Kimenet: $S \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $S = \sum_{i=1}^{N} Jöv_i.be - Jöv_i.ki$

Megjegyzés: értelmezhetjük úgy, hogy

$$\Sigma(Jöv_{1..N}) := \Sigma(Jöv_{1..N-1}) \oplus Jöv_N ,$$

ahol $\oplus : \mathbb{N} \times Nyer \to \mathbb{N}$; $S \oplus F := S + F.be - F.ki$ $\Rightarrow$



Sorozatszámítás (összegzés) tétel.

Sorozatszámítás

rekordok: jövedelem

Így specifikálhatjuk a 6. előadás után.

Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}$,
  $Jöv \in Nyer^N$, $Nyer = Be \times Ki$, $Be, Ki = \mathbb{N}$
- Kimenet: $S \in \mathbb{N}$
- Előfeltétel: –
- ~~Utófeltétel: $S = \sum_{i=1}^{N} Jöv_i.be - Jöv_i.ki$~~

Megjegyzés: értelmezhetjük úgy, hogy

$$\Sigma(Jöv_{1..N}) := \Sigma(Jöv_{1..N-1}) \oplus Jöv_N ,$$

ahol $\oplus : \mathbb{N} \times Nyer \to \mathbb{N}$; $S \oplus F := S + F.be - F.ki$ $\Rightarrow$

- Utófeltétel: $S = \sum_{i=1}^{N} Jöv_i$

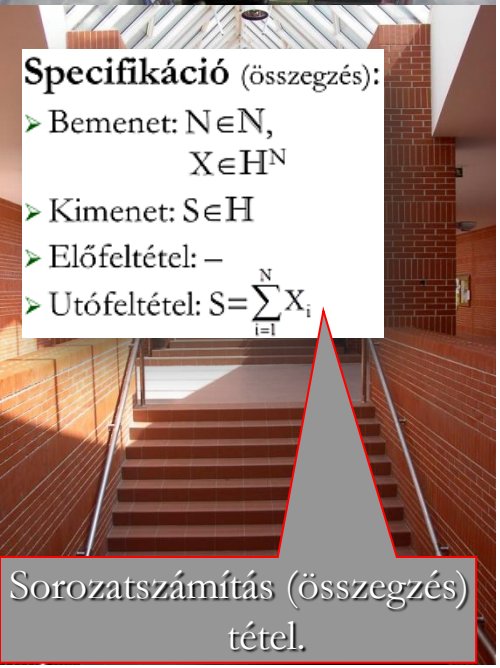
Sorozatszámítás

rekordok: jövedelem

Így specifikálhatjuk a 6. előadás után.

Specifikáció$_2$:

- Bemenet: $N \in \mathbb{N}$,
  $Jöv \in Nyer^N$, $Nyer = Be \times Ki$, $Be, Ki = \mathbb{N}$
- Kimenet: $S \in \mathbb{N}$
- Előfeltétel: –
- Utófeltétel: $S = \sum_{i=1}^{N} Jöv_i$

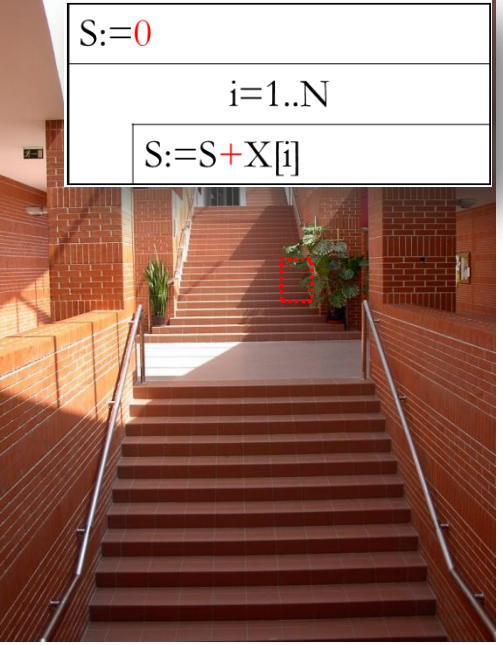


- Definíció:

$\Sigma(Jöv_{1..N}) := \Sigma(Jöv_{1..N-1}) \oplus Jöv_N$;

$\oplus: \mathbb{N} \times Nyer \to \mathbb{N}$; $S \oplus F := S + F.be - F.ki$

S:=0
i=1..N
S:=S+X[i]



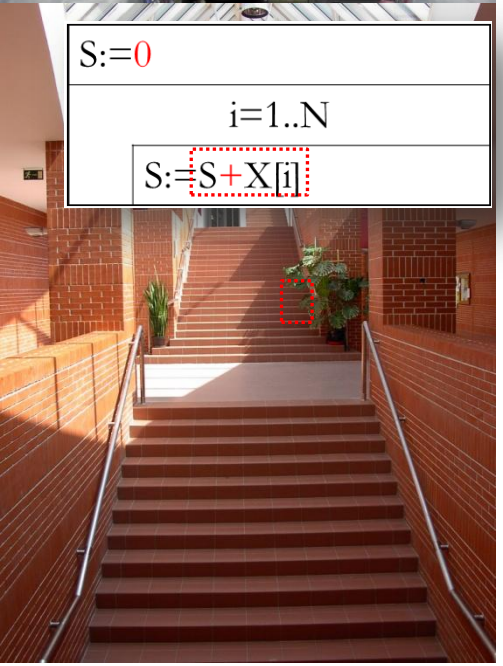
Sorozatszámítás

rekordok: jövedelem

Algoritmus$_1$ – „régi” változat:

S:=0
i=1..N
S:=S+Be[i]–Ki[i]

Változó
i:**Egész**



Sorozatszámítás

rekordok: jövedelem

Algoritmus$_1$ – „régi” változat:

Változó
i:**Egész**

```
S:=0
i=1..N
    S:=S+Be[i]–Ki[i]
```

Algoritmus$_2$ – „új” változat:

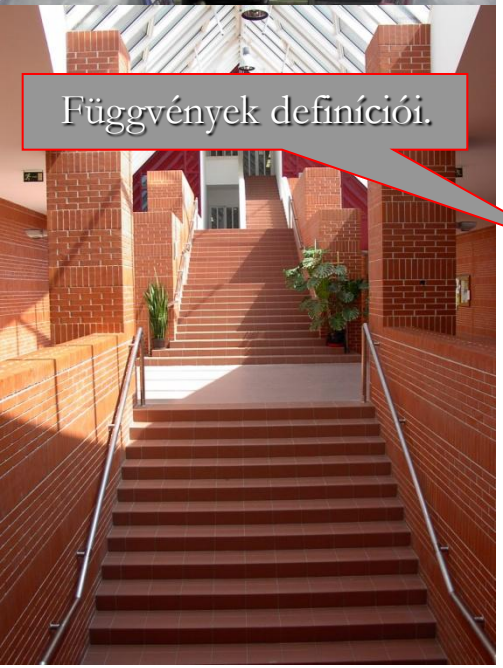
Változó
i:**Egész**

```
S:=0
i=1..N
    S:=S⊕Jöv[i]
```

Megírandó a ⊕ operátor!
S⊕F:=S+F.be–F.ki

Sorozatszámítás
rekordok: jövedelem

Típus- és függvényfej-definíciók.



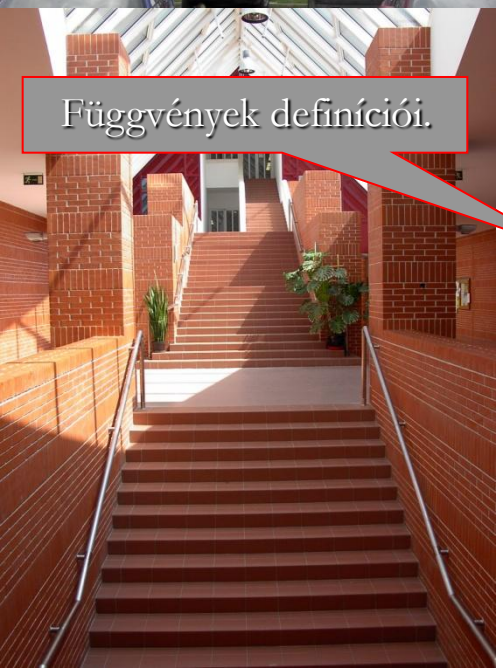
Függvények definíciói.

Kód – operátor (egy „minimum program”):

```
//TNyer típus definiálása:
typedef struct {int be, ki;} TNyer;//reprezentáció
int operator +(int s, TNyer x);//TNyer típusú hozzáadás operátor fejsora
//a lényegi számítás függvénye:
int sum(int n, const TNyer t[]);//n TNyer 'összege' függvény fejsora
//Billentyűre várás:
void billreVar();

int main()
{
  //bemenet -csak most: konstansok, így nem kell beolvasni!-:
  const TNyer Jov[]={{10,5},{5,10},{100,50}};//jövedelem tömb értékei
  int N=sizeof Jov / sizeof(TNyer);//aktuális elemszám
  //kimenet -mindjárt számítással-:
  int S=sum(N,Jov);
  //eredménymegjelenítés:
  cout << "Ossz jovedelem:" << S << endl;

  billreVar();
  return 0;
}

//TNyer típusú hozzáadás operátor definíciója:
int operator +(int s, TNyer x)
{
  return s + x.be - x.ki;
}

//n TNyer 'összege' függvény definíciója:
int sum(int n, const TNyer t[])
{
  int sum=0;
  for (int i=0; i<n; i++)
  {
    sum=sum+t[i];
  }
  return sum;
}
```

Sorozatszámítás
rekordok: jövedelem

Típus- és függvényfej-definíciók.



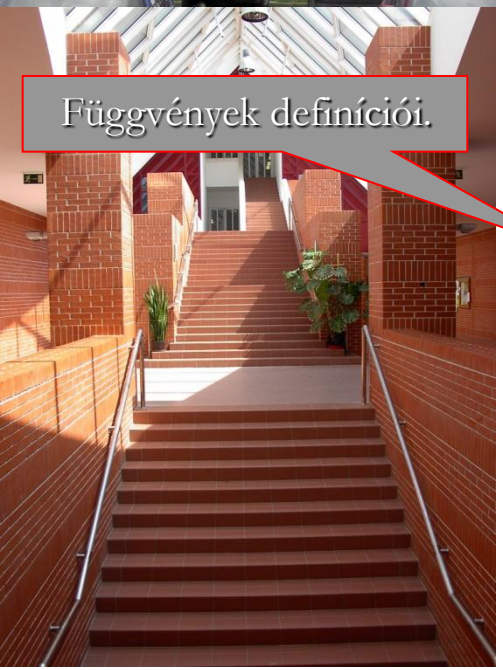
Függvények definíciói.

Kód – operátor (egy „minimum program”):

```
//TNyer típus definiálása:
typedef struct {int be, ki;} TNyer;//reprezentáció
int operator +(int s, TNyer x);//TNyer típusú hozzáadás operátor fejsora
//a lényegi számítás függvénye:
int sum(int n, const TNyer t[]);//n TNyer 'összege' függvény fejsora
//Billentyűre várás:
void billreVar();

int main()
{
  //bemenet -csak most: konstansok, így nem kell beolvasni!-:
  const TNyer Jov[]={{10,5},{5,10},{100,50}};//jövedelem tömb értékei
  int N=sizeof Jov / sizeof(TNyer);//aktuális elemszám
  //kimenet -mindjárt számítással-:
  int S=sum(N,Jov);
  //eredménymegjelenítés:
  cout << "Ossz jovedelem:" << S << endl;

  billreVar();
  return 0;
}

//TNyer típusú hozzáadás operátor definíciója:
int operator +(int s, TNyer x)
{
  return s + x.be - x.ki;
}

//n TNyer 'összege' függvény definíciója:
int sum(int n, const TNyer t[])
{
  int sum=0;
  for (int i=0; i<n; i++)
  {
    sum=sum+t[i];
  }
  return sum;
}
```

Kód jegyzetként

Sorozatszámítás
rekordok: jövedelem



Típus- és függvényfej-definíciók.

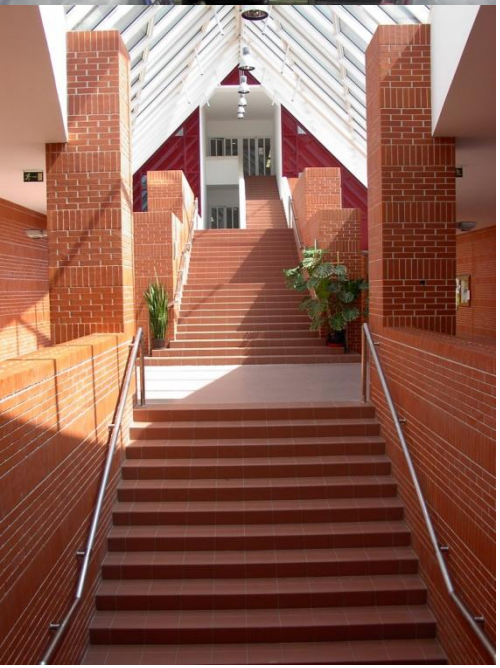
Függvények definíciói.

Kód – operátor (egy „minimum program”):

```
//TNyer típus definiálása:
typedef struct {int be, ki;} TNyer;//reprezentáció
int operator +(int s, TNyer x);//TNyer típusú hozzáadás operátor fejsora
//a lényegi számítás függvénye:
int sum(int n, const TNyer t[]);//n TNyer 'összege' függvény fejsora
//Billentyűre várás:
void billreVar();

int main()
{
  //bemenet -csak most: konstansok, így nem kell beolvasni!-:
  const TNyer Jov[]={{10,5},{5,10},{100,50}};//jövedelem tömb értékei
  int N=sizeof Jov / sizeof(TNyer);//aktuális elemszám
  //kimenet -mindjárt számítással-:
  int S=sum(N,Jov);
  //eredménymegjelenítés:
  cout << "Ossz jovedelem:" << S << endl;

  billreVar();
  return 0;
}

//TNyer típusú hozzáadás operátor definíciója:
int operator +(int s, TNyer x)
{
  return s + x.be - x.ki;
}

//n TNyer 'összege' függvény definíciója:
int sum(int n, const TNyer t[])
{
  int sum=0;
  for (int i=0; i<n; i++)
  {
    sum=sum+t[i];
  }
  return sum;
}
```

De jó lenne, ha nem kellene újraírni a sum függvényt csak amiatt, h. más típusú elemeket kell összeadni!

Kód jegyzet-ként



Maximum

rekordok: legkorábbi születésnap

Feladat:

Melyik a legkorábbi a születésnap?

Maximum

rekordok: legkorábbi születésnap

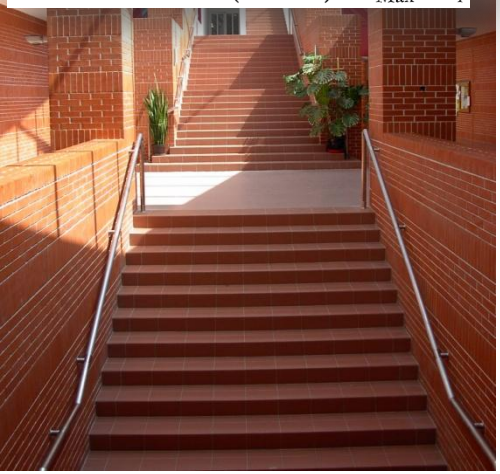
Feladat:

Melyik a legkorábbi a születésnap?

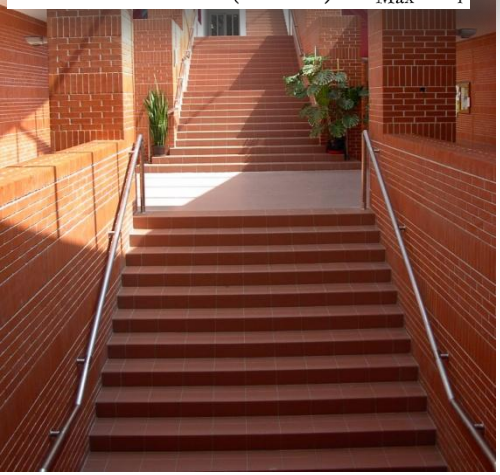
Specifikáció$_1$:

Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N > 0$
- Utófeltétel: $1 \leq Max \leq N$ és $\forall i\ (1 \leq i \leq N): X_{Max} \geq X_i$



Specifikáció:
- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N): X_{Max} \ge X_i$



Maximum

rekordok: legkorábbi születésnap

Feladat:

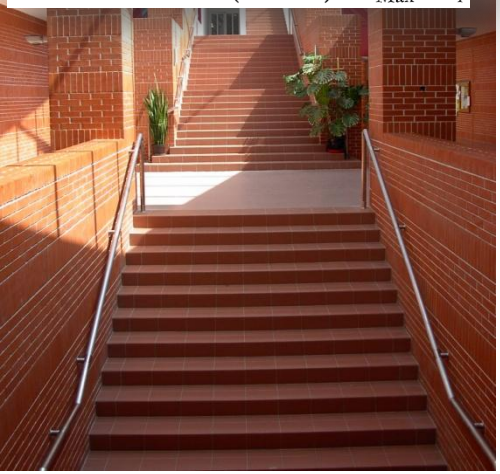
Melyik a legkorábbi a születésnap?

Specifikáció$_1$:

- Bemenet: $N \in \mathbf{N}$, $Hó,Nap \in \mathbf{Z}^N$
- Kimenet: $Min \in \mathbf{N}$
- Előfeltétel: $N>0$ …
- Utófeltétel: $1 \le Min \le N$ és

$$\forall i(1 \le i \le N): \quad Hó_{Min} < Hó_i \text{ vagy}$$
$$Hó_{Min} = Hó_i \text{ és}$$
$$Nap_{Min} \le Nap_i$$

Specifikáció:
- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N)$: $X_{Max} \ge X_i$



Maximum

rekordok: legkorábbi születésnap

Specifikáció$_2$:

- Bemenet: $N \in \mathbf{N}$,
  $Szül \in Dátum^N$,
  $Dátum = Hó \times Nap$, $Hó, Nap = \mathbf{N}$

Maximum

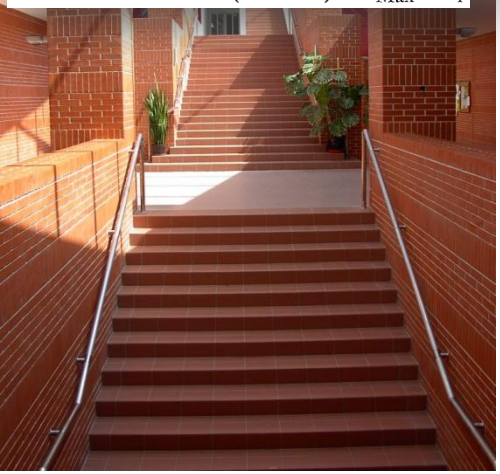
rekordok: legkorábbi születésnap

Specifikáció:
- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N)$: $X_{Max} \ge X_i$

Specifikáció$_2$:

- Bemenet: $N \in \mathbf{N}$, $Szül \in Dátum^N$, $Dátum = Hó \times Nap$, $Hó, Nap = \mathbf{N}$
- Kimenet: $Min \in \mathbf{N}$
- Előfeltétel: $N>0$ …
- Utófeltétel: $1 \le Min \le N$ és $\forall i(1 \le i \le N)$: $Szül_{Min} \le Szül_i$

Specifikáció:
- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N): X_{Max} \ge X_i$

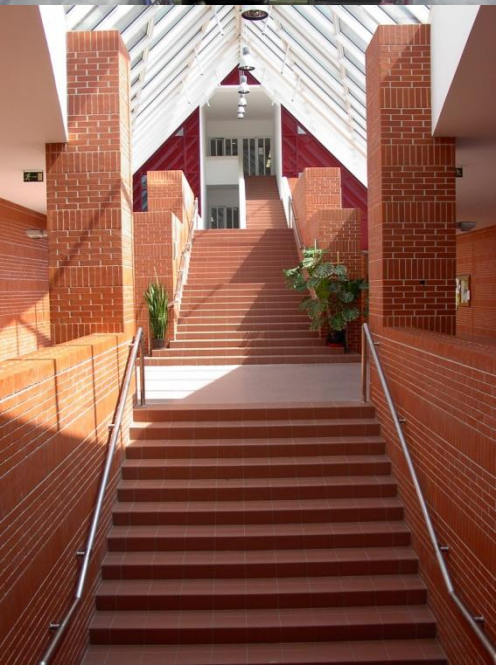


Maximum

rekordok: legkorábbi születésnap

Specifikáció$_2$:

- Bemenet: $N \in \mathbf{N}$,
  $Szül \in Dátum^N$,
  $Dátum = Hó \times Nap$, $Hó, Nap = \mathbf{N}$
- Kimenet: $Min \in \mathbf{N}$
- Előfeltétel: $N>0$ …
- Utófeltétel: $1 \le Min \le N$ és
  $\forall i(1 \le i \le N): Szül_{Min} \le Szül_i$
- Definíció:
  $\le: Dátum \times Dátum \rightarrow \mathbf{L}$
  $d1 \le d2 := d1.hó < d2.hó$ vagy
  $d1.hó = d2.hó$ és $d1.nap \le d2.nap$



Maximum

rekordok: legkorábbi születésnap

Specifikáció$_2$:

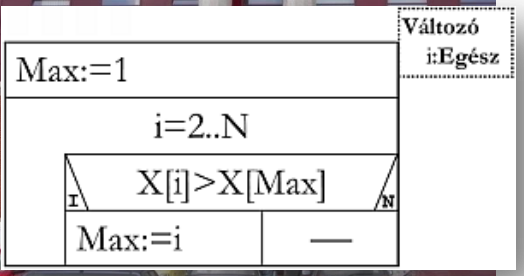
- Bemenet: $N \in \mathbf{N}$,
  $Szül \in Dátum^N$,
  $Dátum = Hó \times Nap$, $Hó, Nap = \mathbf{N}$
- Kimenet: $Min \in \mathbf{N}$
- Előfeltétel: $N > 0$ …
- Utófeltétel: $Min = \underset{i=1}{\overset{N}{Min}} Szül[i]$
- Definíció:

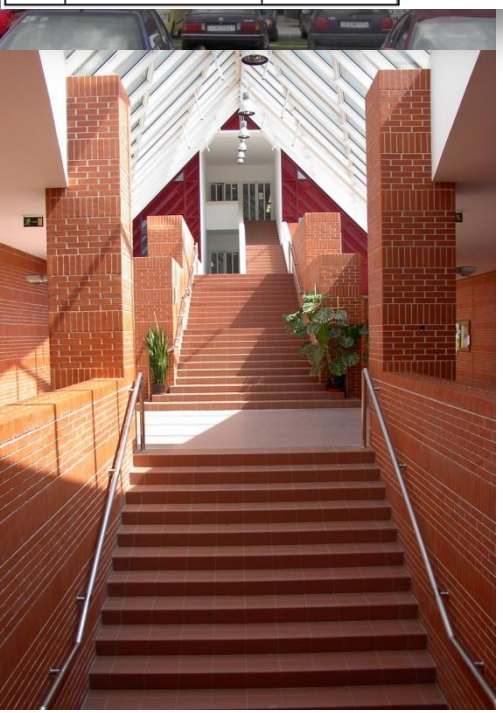
  $\leq : Dátum \times Dátum \rightarrow \mathbf{L}$

  $d1 \leq d2 := d1.hó < d2.hó$ vagy
  $d1.hó = d2.hó$ és $d1.nap \leq d2.nap$

Maximum

rekordok: legkorábbi születésnap





Változó
i:Egész

Min:=1

i=2..N

Szül[i]<Szül[Min]

I N

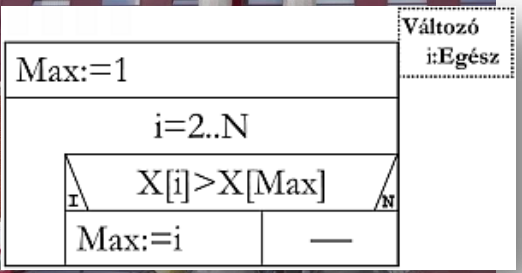
Min:=i | —

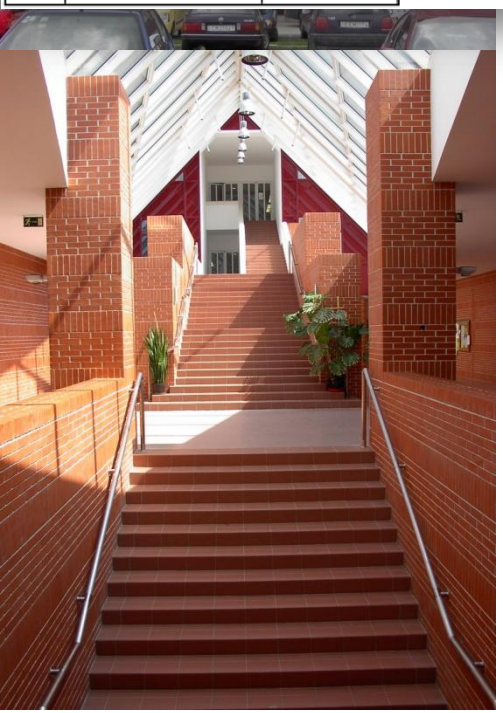
Természetesen meg kell még írni az **TDátum** típusra a < relációt megvalósító függvényt (operátort).

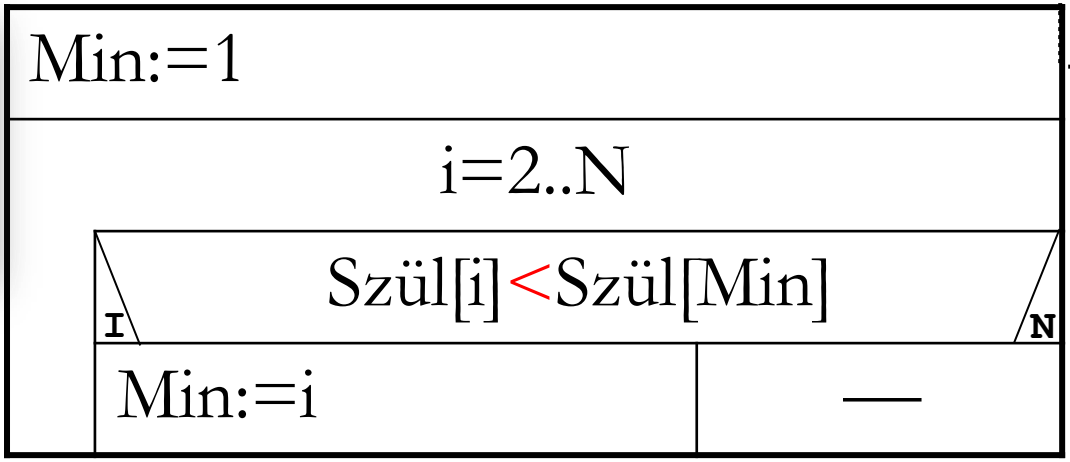
Az világos, hogy a specifikációban felbukkanó ≤ és az algoritmusbeli < a relációk egymással kifejezhetők.

Maximum

rekordok: legkorábbi születésnap





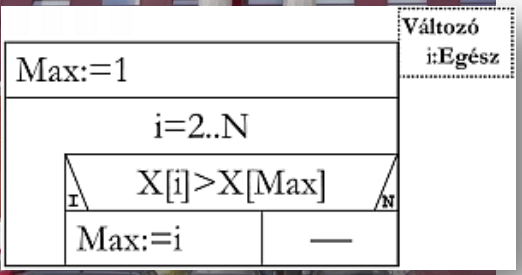


Természetesen meg kell még írni az **TDátum** típusra a < relációt megvalósító függvényt (operátort).

Az világos, hogy a specifikációban felbukkanó ≤ és az algoritmusbeli < a relációk egymással kifejezhetők.

Kóddarab jegyzetként

Maximum
rekordok: legkorábbi születésnap



Változó
i:Egész

Min:=1

i=2..N

Szül[i]<Szül[Min]

I N

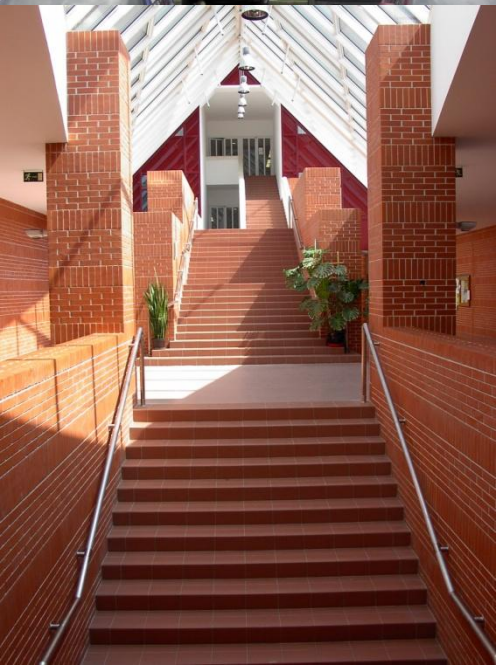
Min:=i | —

De jó lenne, ha nem kellene újraírni a MinInd függvényt csak amiatt, h. mástípusú elemeket kell összehasonlítani!

Természetesen meg kell még írni az **TDátum** típusra a < relációt megvalósító függvényt (operátort).

Az világos, hogy a specifikációban felbukkanó ≤ és az algoritmusbeli < a relációk egymással kifejezhetők.

Kóddarab jegyzetként

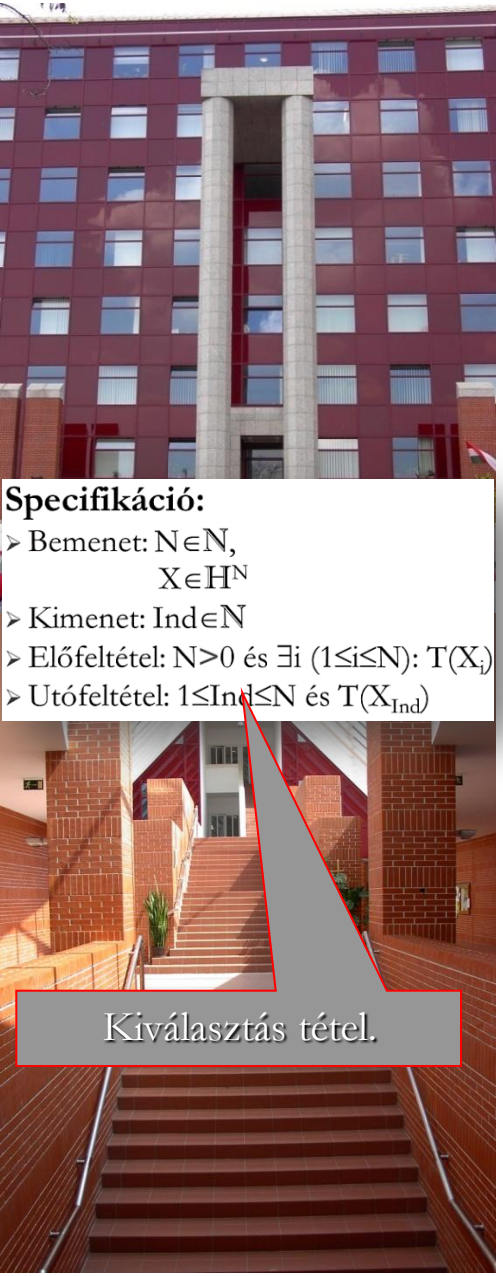


Függvény a feltételben

Feladat:

Adjunk meg egy magánhangzót egy magyar szóban!

Függvény a feltételben



Feladat:

Adjunk meg egy magánhangzót egy magyar szóban!

Specifikáció: (kiválasztás)

- Bemenet: $N \in \mathbb{N}$, $Szó \in \mathbb{K}^N$
- Kimenet: $Mh \in \mathbb{N}$
- Előfeltétel: $N>0$ és $\exists i(1 \le i \le N)$: $MagánhangzóE(Szó_i)$
- Utófeltétel: $1 \le Mh \le N$ és $MagánhangzóE(Szó_{Mh})$

Függvény a feltételben

Feladat:

Adjunk meg egy magánhangzót egy magyar szóban!

Specifikáció: (kiválasztás)

- Bemenet: $N \in \mathbb{N}$, $Szó \in K^N$
- Kimenet: $Mh \in \mathbb{N}$
- Előfeltétel: $N>0$ és $\exists i (1 \le i \le N)$: MagánhangzóE($Szó_i$)
- Utófeltétel: $1 \le Mh \le N$ és MagánhangzóE($Szó_{Mh}$)

Másképpen: $Mh = \text{Kiválaszt}_{i=1}^{N}(\text{MagánhangzóE}(Szó_i))$

Függvény a feltételben

Feladat:

Adjunk meg egy magánhangzót egy magyar szóban!

Specifikáció: (kiválasztás)

Lehetne: Szó∈S

- Bemenet: $N \in \mathbb{N}$, $Szó \in \mathbb{K}^N$
- Kimenet: $Mh \in \mathbb{N}$
- Előfeltétel: $N>0$ és $\exists i (1 \le i \le N)$: MagánhangzóE($Szó_i$)
- Utófeltétel: $1 \le Mh \le N$ és MagánhangzóE($Szó_{Mh}$)

Másképpen: $Mh = \text{Kiválaszt}_{i=1}^{N}(\text{MagánhangzóE}(Szó_i))$

- Definíció:

$\text{MagánhangzóE}: \mathbb{K} \to \mathbb{L}$

$\text{MagánhangzóE}(B) = \exists i (1 \le i \le 14): B = Mag_i$

$Mag \in \mathbb{K}^{14} = (\text{"a"}, \text{"á"}, \ldots, \text{"ű"})$

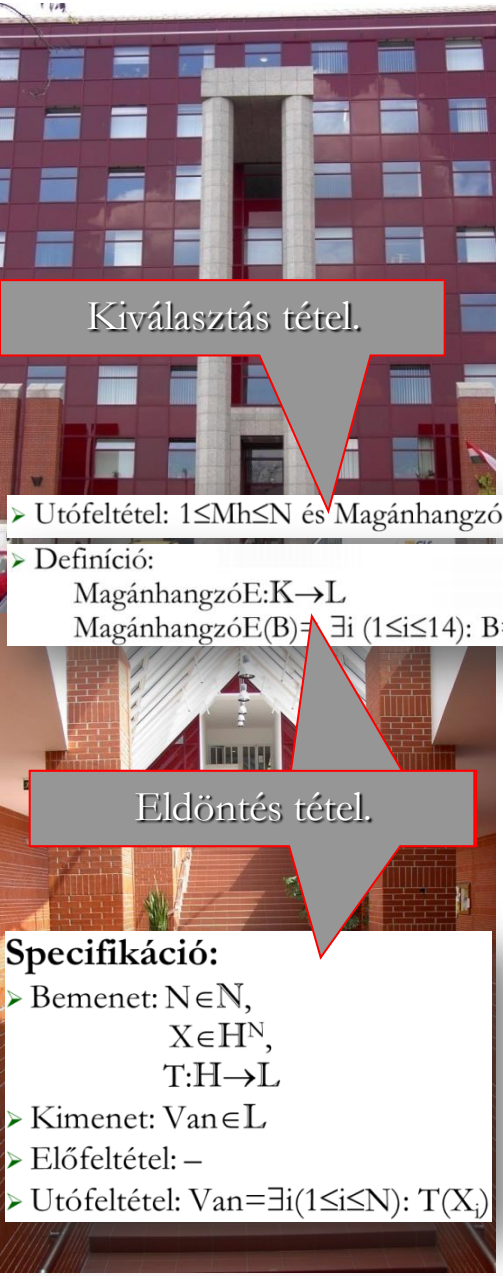
Eldöntés tétel.

Függvény a feltételben

Algoritmus:

```
Mh:=1
nem MagánhangzóE(Szó[i])
    Mh:=Mh+1
```

Függvény a feltételben



Kiválasztás tétel.

- Utófeltétel: $1 \le Mh \le N$ és MagánhangzóE($Szó_{Mh}$)
- Definíció:
  MagánhangzóE:K→L
  MagánhangzóE(B)⇔ ∃i (1≤i≤14): B=Mag_i

Eldöntés tétel.

Specifikáció:

- Bemenet: N∈N,
  X∈H^N,
  T:H→L
- Kimenet: Van∈L
- Előfeltétel: –
- Utófeltétel: Van=∃i(1≤i≤N): T(X_i)

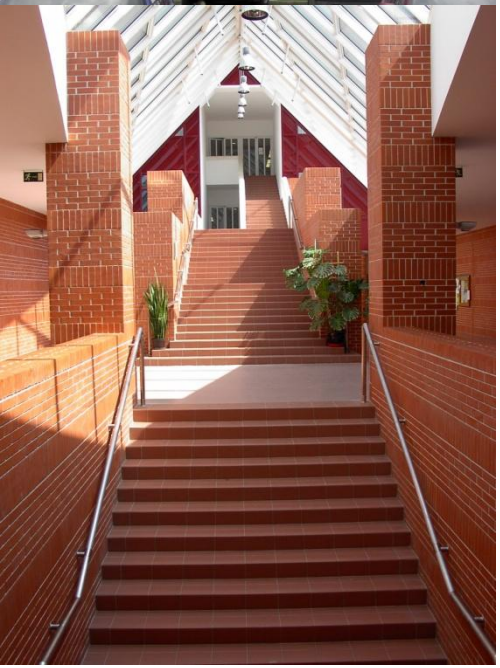
Algoritmus:

```
Mh:=1
nem MagánhangzóE(Szó[i])
    Mh:=Mh+1
```

MagánhangzóE(B:**Karakter**):**Logikai**

```
i:=1
i≤14 és B≠Mag[i]
    i:=i+1
MagánhangzóE:=i≤14
```

Változó
i:**Egész**
Konstans
Mag:Tön

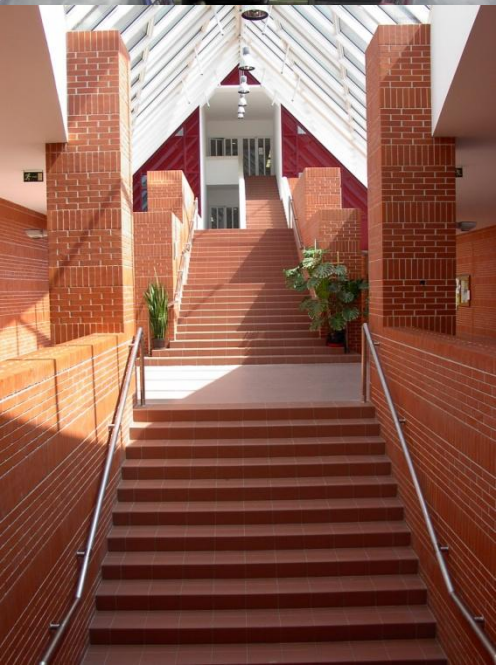


Mátrixok

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére pontsokszorozással: minden régi pont helyébe 2*2 azonos színű pontot raj-zolunk a nagyított képen.

→



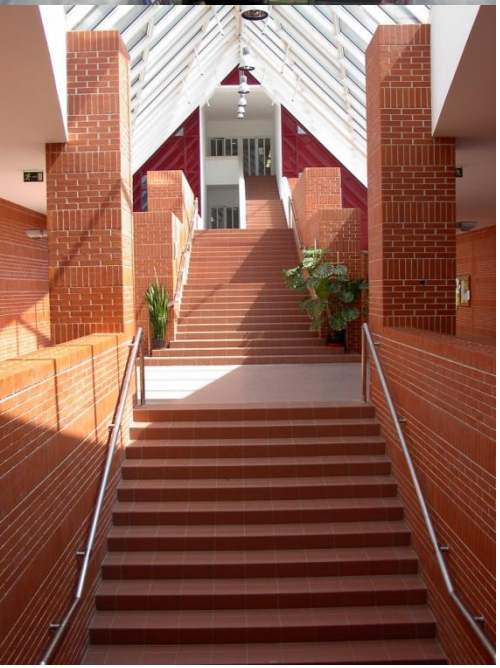
Mátrixok

Problémák/válaszok:

- Hogyan ábrázoljunk egy képet?
  A kép rendezett pontokból áll, azaz biztosan valamilyen sorozatként adható meg.
- Nehézkes lenne azonban a pontokra egy sorszámozást adni.
  Kézenfekvőbb azt megmondani, hogy egy képpont a kép hányadik sorában, illetve oszlopában található, azaz alkalmazzunk **dupla indexelés**t!
  A kétindexes tömböket hívjuk **mátrix**nak.

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére *pontsokszorozás*sal: minden régi pont helyébe 2*2 azonos színű pontot rajzolunk a nagyított képen.



Mátrixok

:=$(\mathbb{N}^M)^N$

N – a sorok,

M – az oszlopok száma

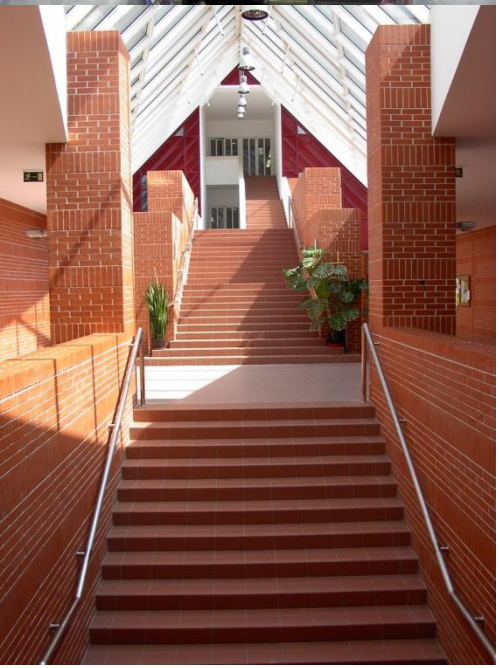
Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$

Mátrixok

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére *pontsokszorozás*sal: minden régi pont helyébe 2*2 azonos színű pontot rajzolunk a nagyított képen.



:=$(\mathbb{N}^M)^N$

N – a sorok,

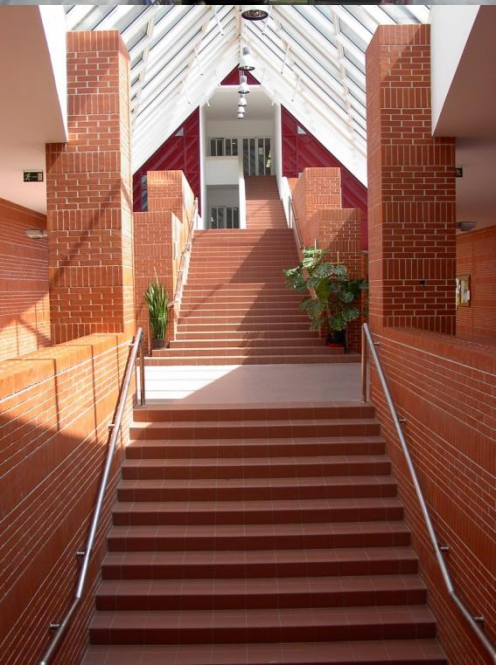
M – az oszlopok száma

Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $NK \in \mathbb{N}^{2 \cdot N \times 2 \cdot M}$

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére *pontsokszorozás*sal: minden régi pont helyébe 2*2 azonos színű pontot rajzolunk a nagyított képen.



Mátrixok

:=$(\mathbb{N}^M)^N$

N – a sorok,

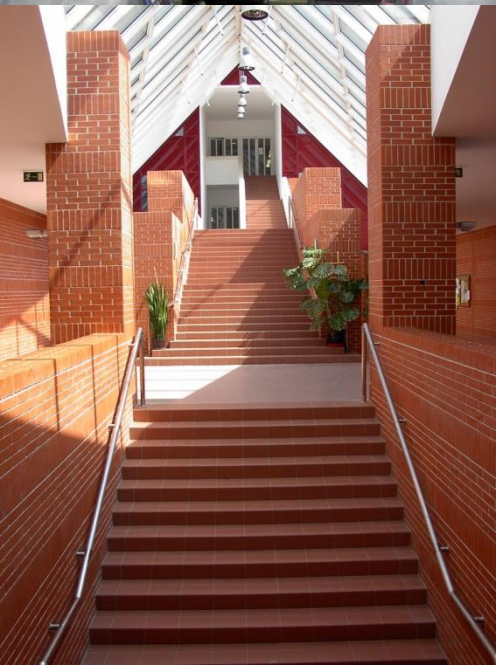
M – az oszlopok száma

Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $NK \in \mathbb{N}^{2 \cdot N \times 2 \cdot M}$
- Előfeltétel: –

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére *pontsokszorozás*sal: minden régi pont helyébe 2*2 azonos színű pontot rajzolunk a nagyított képen.



Mátrixok

:=$(\mathbb{N}^M)^N$

N – a sorok,

M – az oszlopok száma

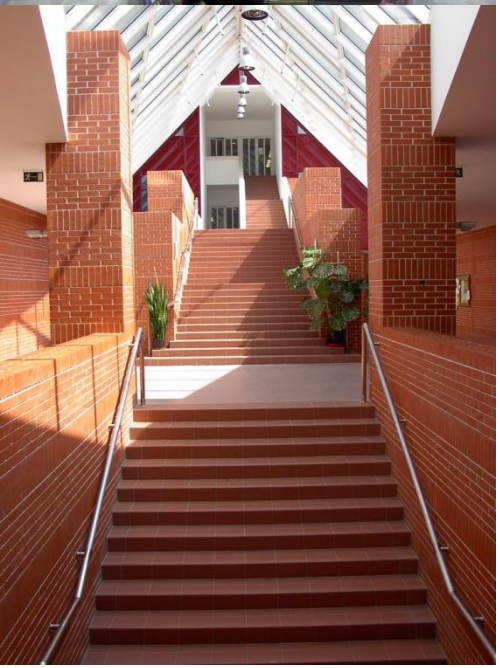
Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $NK \in \mathbb{N}^{2 \cdot N \times 2 \cdot M}$
- Előfeltétel: –
- Utófeltétel: $\forall i (1 \le i \le N)$: $\forall j (1 \le j \le M)$:

$$NK_{2 \cdot i, 2 \cdot j} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i-1, 2 \cdot j} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i, 2 \cdot j-1} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i-1, 2 \cdot j-1} = K_{i,j}$$

Feladat:

Egy N*M-es raszterképet nagyítsunk a két-szeresére *pontsokszorozás*sal: minden régi pont helyébe 2*2 azonos színű pontot rajzolunk a nagyított képen.



Mátrixok

:=$(\mathbb{N}^M)^N$

N – a sorok,

M – az oszlopok száma

Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $NK \in \mathbb{N}^{2 \cdot N \times 2 \cdot M}$
- Előfeltétel: –
- Utófeltétel: $\forall i (1 \le i \le N)$: $\forall j (1 \le j \le M)$:

$$NK_{2 \cdot i, 2 \cdot j} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i-1, 2 \cdot j} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i, 2 \cdot j-1} = K_{i,j} \text{ és}$$
$$NK_{2 \cdot i-1, 2 \cdot j-1} = K_{i,j}$$

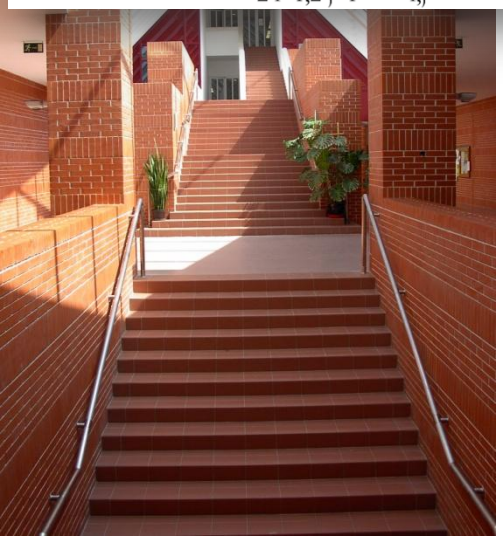
Ez a **másolás** tétel egy variációja, csak egy elemből négy elem keletkezik.

Mátrixok

Algoritmus:

Specifikáció:

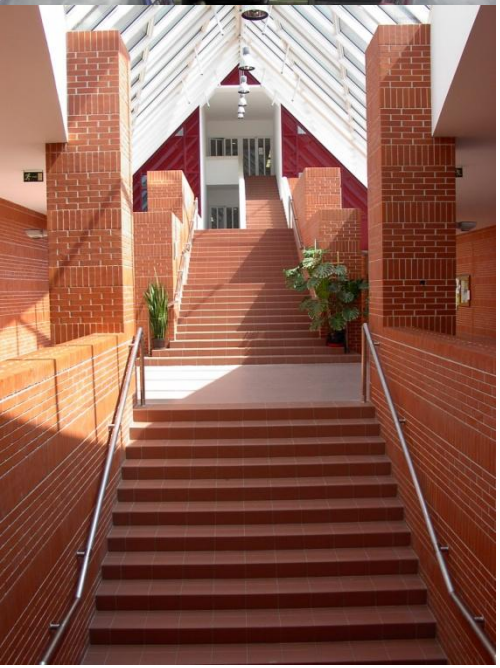
- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $NK \in \mathbb{N}^{2 \cdot N \times 2 \cdot M}$
- Előfeltétel: –
- Utófeltétel: $\forall i\ (1 \le i \le N)$: $\forall j\ (1 \le j \le M)$:
  $NK_{2 \cdot i, 2 \cdot j} = K_{i,j}$ és
  $NK_{2 \cdot i-1, 2 \cdot j} = K_{i,j}$ és
  $NK_{2 \cdot i, 2 \cdot j-1} = K_{i,j}$ és
  $NK_{2 \cdot i-1, 2 \cdot j-1} = K_{i,j}$



Változó
i,j:**Egész**

```
i=1..N
    j=1..M
        NK[2*i,2*j]:=K[i,j]
        NK[2*i–1,2*j]:=K[i,j]
        NK[2*i,2*j–1]:=K[i,j]
        NK[2*i–1,2*j–1]:=K[i,j]
```

Megjegyzés: programozási nyelvekben a mátrix elemének elérésére más jelölés is lehet, pl.: C++ esetén K[i][j].

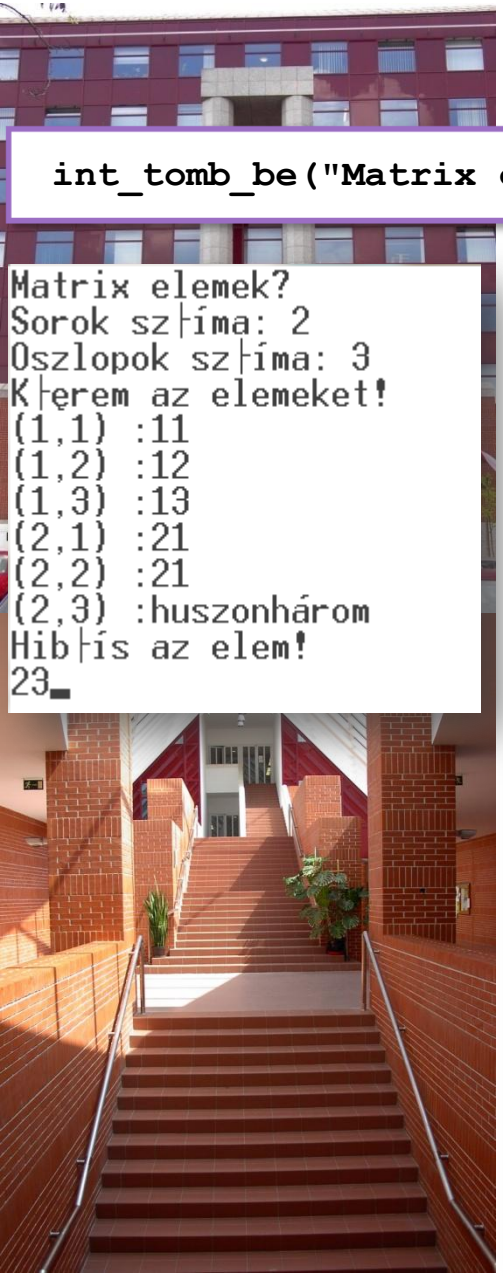


Mátrixok

Egy mátrix kódolási példa:

Írjunk mátrix-beolvasó eljárást!

Mátrixok



```
int_tomb_be("Matrix elemek?",…);
```

```
Matrix elemek?
Sorok szíma: 2
Oszlopok szíma: 3
Kérem az elemeket!
(1,1) :11
(1,2) :12
(1,3) :13
(2,1) :21
(2,2) :21
(2,3) :huszonhárom
Hibás az elem!
23
```

Egy mátrix kódolási példa:

Írjunk mátrix-beolvasó eljárást!

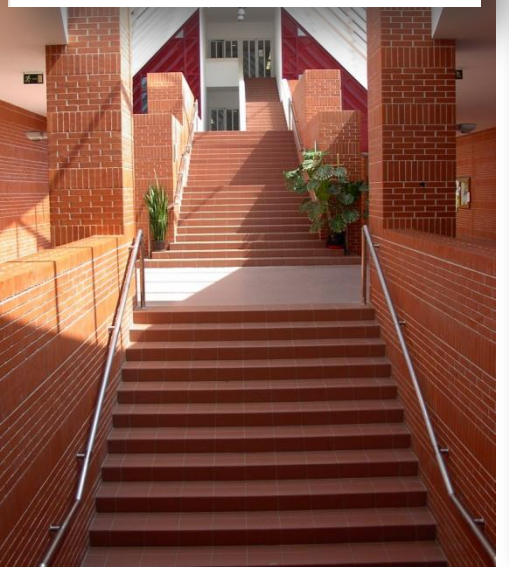
Megoldás:

```
//egy (min,max) közötti értékű, int elemekből álló,
//(maxN, maxM) méretekkel deklarált mat mátrix beolvasása billentyűzetről:
void int_tomb_be(string kerd, int min, int max, int mat[maxN][maxM],
                 int &n, int &m, int maxN, int maxM)
{
    cout << kerd << endl;
    //tényleges méretek beolvasása:
    be_int("Sorok száma: ",n,0,maxN,"Hibás a sorok száma!");
    be_int("Oszlopok száma: ",m,0,maxM,"Hibás az oszlopok száma!");
    //a mátrix beolvasása:
    cout << "Kérem az elemeket!" << endl;
    for (int i=1;i<=n;++i)
    {
      for (int j=1;j<=m;++j)
      {
        cout << "(" << i << "," << j << ") :";
        be_int("",mat[i-1][j-1],min,max,"Hibás az elem!");
      }
    }
}
```

Mátrixok

```
int_tomb_be("Matrix elemek?",…);
```

```
Matrix elemek?
Sorok szíma: 2
Oszlopok szíma: 3
Kérem az elemeket!
(1,1) :11
(1,2) :12
(1,3) :13
(2,1) :21
(2,2) :21
(2,3) :huszonhárom
Hibás az elem!
23
```



Egy mátrix kódolási példa:

Írjunk mátrix-beolvasó eljárást!

Megoldás:

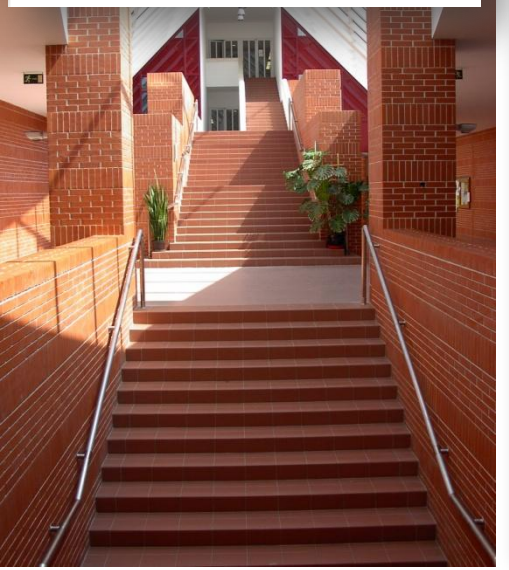
```
//egy (min,max) közötti értékű, int elemekből álló,
//(maxN, maxM) méretekkel deklarált mat mátrix beolvasása billentyűzetről:
void int_tomb_be(string kerd, int min, int max, int mat[maxN][maxM],
                 int &n, int &m, int maxN, int maxM)
{
    cout << kerd << endl;
    //tényleges méretek beolvasása:
    be_int("Sorok száma: ",n,0,maxN,"Hibás a sorok száma!");
    be_int("Oszlopok száma: ",m,0,maxM,"Hibás az oszlopok száma!");
    //a mátrix beolvasása:
    cout << "Kérem az elemeket!" << endl;
    for (int i=1;i<=n;++i)
    {
      for (int j=1;j<=m;++j)
      {
        cout << "(" << i << "," << j << ") :";
        be_int("",mat[i-1][j-1],min,max,"Hibás az elem!");
      }
    }
}
```

Kód jegyzetként

Mátrixok

```
int_tomb_be("Matrix elemek?",…);
```

```
Matrix elemek?
Sorok szíma: 2
Oszlopok szíma: 3
Kérem az elemeket!
(1,1) :11
(1,2) :12
(1,3) :13
(2,1) :21
(2,2) :21
(2,3) :huszonhárom
Hibás az elem!
23
```



Egy mátrix kódolási példa:

Írjunk mátrix-beolvasó eljárást!

Megoldás:

De jó lenne, ha nem kellene új függvényt írni float vagy … vagy string (?) elemű tömbhöz!

```
//egy (min,max) közötti értékű, int elemekből álló,
//(maxN, maxM) méretekkel deklarált mat mátrix beolvasása billentyűzetről:
void int_tomb_be(string kerd, int min, int max, int mat[maxN][maxM],
                 int &n, int &m, int maxN, int maxM)
{
    cout << kerd << endl;
    //tényleges méretek beolvasása:
    be_int("Sorok száma: ",n,0,maxN,"Hibás a sorok száma!");
    be_int("Oszlopok száma: ",m,0,maxM,"Hibás az oszlopok száma!");
    //a mátrix beolvasása:
    cout << "Kérem az elemeket!" << endl;
    for (int i=1;i<=n;++i)
    {
      for (int j=1;j<=m;++j)
      {
        cout << "(" << i << "," << j << ") :";
        be_int("",mat[i-1][j-1],min,max,"Hibás az elem!");
      }
    }
}
```

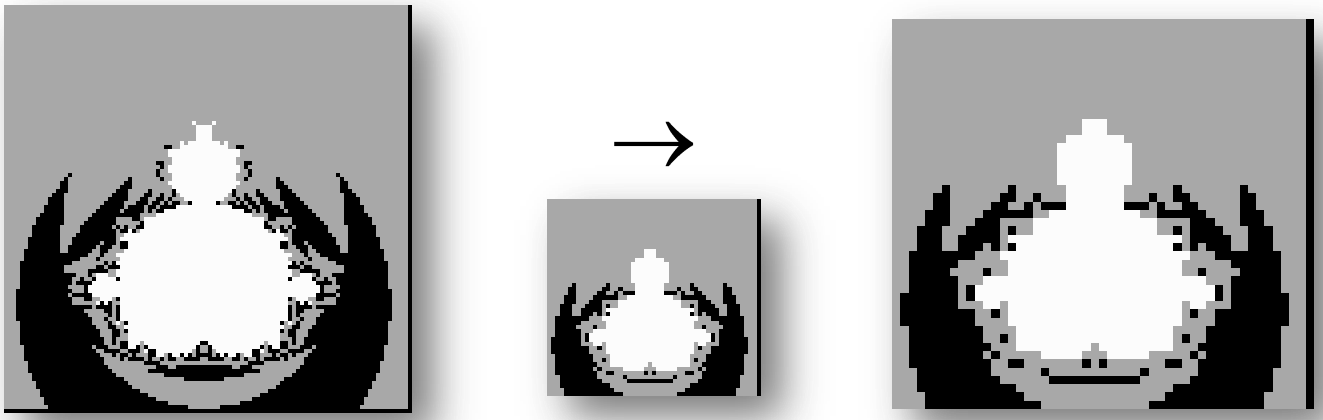
Kód jegyzetként

Mátrixok

Feladat:

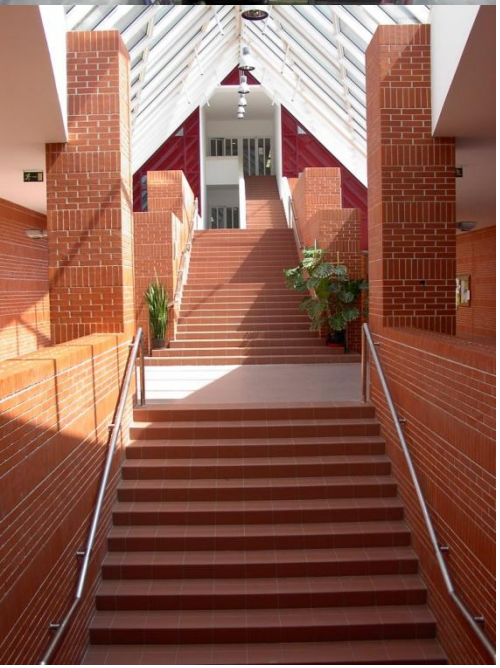
Egy N*M-es raszterképet kicsinyítsünk a felére (N/2*M/2 méretűre) pontátlagolással: a kicsinyített kép minden pontja az eredeti kép 2*2 pontjának „átlaga” legyen!

„átlag”: színkódok átlaga



Feladat:

Egy N*M-es raszterképet kicsinyítsünk a felére (N/2*M/2 méretűre) *pontátlagolás*sal: a kicsinyített kép minden pontja az eredeti kép 2*2 pontjának „átlaga” legyen!



Mátrixok

Specifikáció: (másolás)

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$
- Kimenet: $KK \in \mathbb{N}^{N/2 \times M/2}$
- Előfeltétel: PárosE(N) és PárosE(M)
- Utófeltétel:

$$\forall i(1 \le i \le N/2): \forall j(1 \le j \le M/2):$$
$$KK_{i,j} = (K_{2*i,2*j} + K_{2*i-1,2*j} + K_{2*i,2*j-1} + K_{2*i-1,2*j-1})/4$$

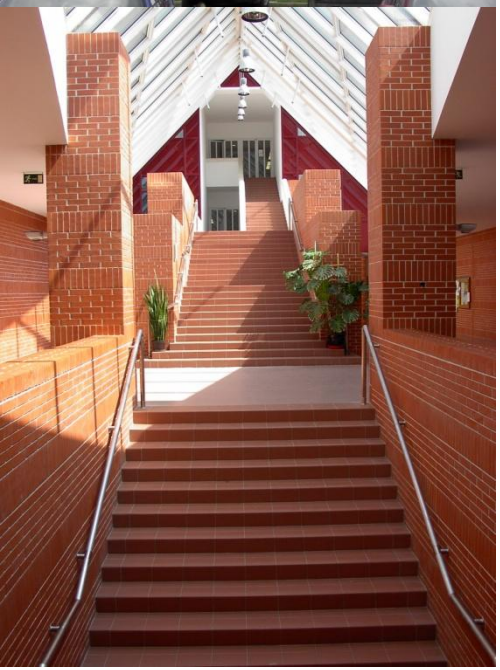
- Definíció:

$$\text{PárosE}: \mathbb{N} \to \mathbb{L}$$
$$\text{PárosE}(x) := x \text{ Mod } 2 = 0$$

Mátrixok

- Utófeltétel:

$$\forall i(1 \le i \le N/2): \forall j(1 \le j \le M/2): KK_{i,j} = (K_{2*i,2*j} + K_{2*i-1,2*j} + K_{2*i,2*j-1} + K_{2*i-1,2*j-1})/4$$

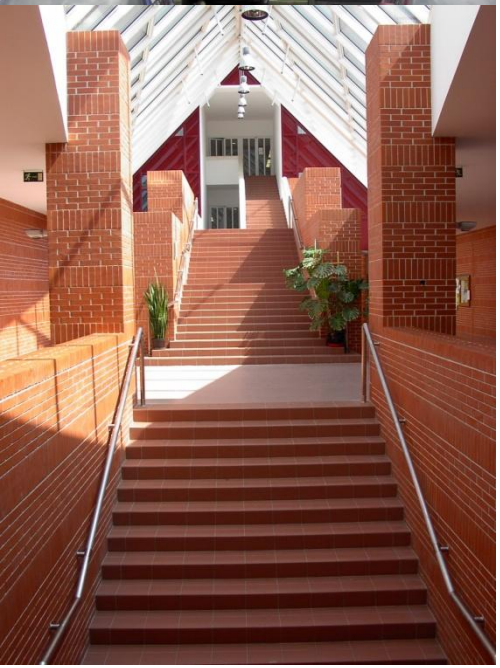


Algoritmus:

Változó
i,j:Ege

```
i=1..N/2
    j=1..M/2
        KK[i,j]:=(K[2*i,2*j]+K[2*i–1,2*j]+
                  K[2*i,2*j–1]+K[2*i–1,2*j–1])/4
```

Megjegyzés: a színes képeknél az átlagolással baj lehet! Milyen szín egy piros és egy kék színű pont átlaga? RGB esetén a szín: Rekord(piros,zöld,kék:**Egész**);
és az átlag?



Mátrixok

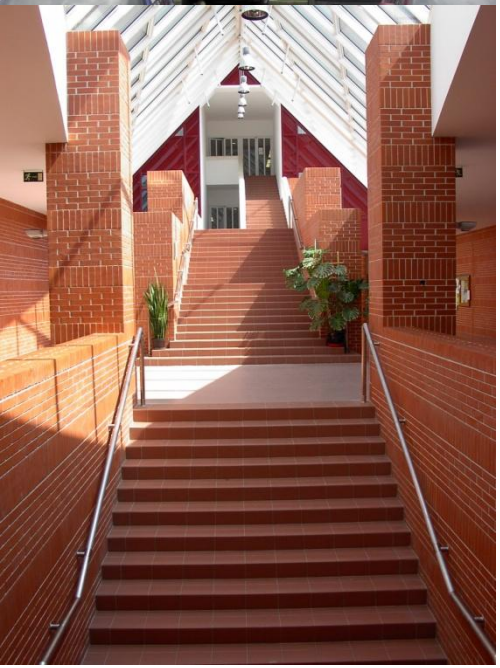
Feladat:

A Rák-köd képére alkalmazzunk egyféle Rank-szűrőt! Minden pontot helyettesítsünk magának és a 8 szomszédjának maximumá-val!

→

Feladat:

A Rák-köd képére alkalmazzunk egyféle *Rank-szűrőt*! Minden pontot helyettesítsünk magának és a 8 szomszédjának maximumával!



Mátrixok

Specifikáció: (maximum-kiválasztás)

- Bemenet: $N,M \in \mathbb{N},\ K \in \mathbb{N}^{N \times M}$
- Kimenet: $RK \in \mathbb{N}^{N \times M}$
- Előfeltétel: –
- Utófeltétel: $\forall i(1<i<N):\ \forall j(1<j<M):$

$$RK_{i,j} = \max_{p=i-1}^{i+1} \max_{q=j-1}^{j+1} K_{p,q} \quad \text{és}$$

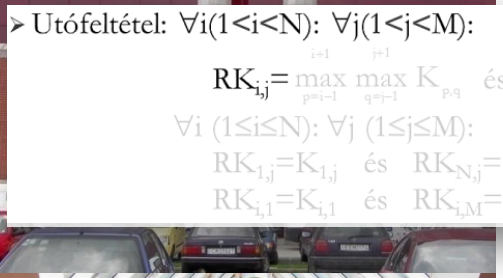
$\forall j(1 \le j \le M):$

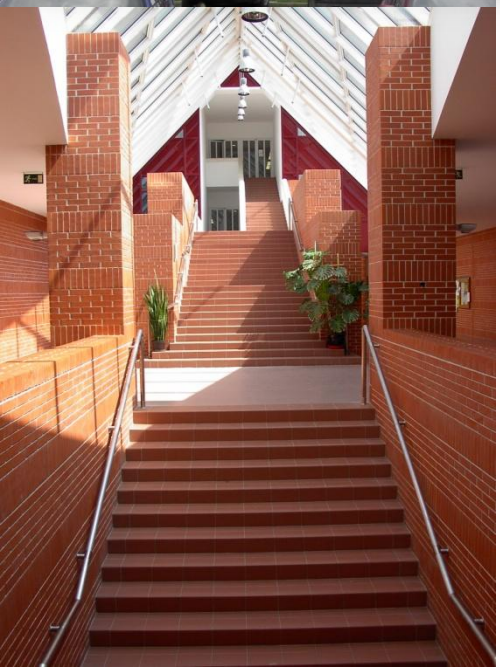
$$RK_{1,j} = K_{1,j} \quad \text{és} \quad RK_{N,j} = K_{N,j}$$

$\forall i(1 \le i \le N):$

$$RK_{i,1} = K_{i,1} \quad \text{és} \quad RK_{i,M} = K_{i,M}$$

Mátrixok





Algoritmus:

- Utófeltétel: $\forall i(1<i<N): \forall j(1<j<M):$

$$RK_{i,j} = \max_{p=i-1}^{i+1} \max_{q=j-1}^{j+1} K_{p,q} \quad \text{és}$$

$\forall i\ (1 \le i \le N): \forall j\ (1 \le j \le M):$

$RK_{1,j}=K_{1,j}$ és $RK_{N,j}=K_{N,j}$

$RK_{i,1}=K_{i,1}$ és $RK_{i,M}=K_{i,M}$

Változó
i,j:Egész

```
i=2..N–1
  j=2..M–1
    Max:=0
      p=i–1..i+1
        q=j–1..j+1
          K[p,q]>Max
          I        N
          Max:=K[p,q]   —
    RK[i,j]:=Max
...
```

Mátrixok

Algoritmus:

- Utófeltétel: $\forall i(1<i<N): \forall j(1<j<M):$

$$RK_{i,j} = \max_{p=i-1}^{i+1} \max_{q=j-1}^{j+1} K_{p,q} \quad \text{és}$$

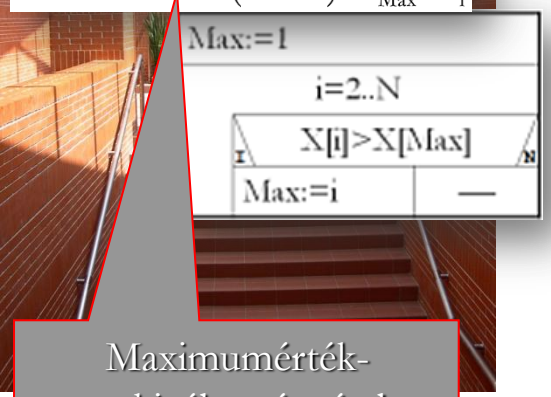
$\forall i\ (1 \le i \le N): \forall j\ (1 \le j \le M):$

$RK_{1,j}=K_{1,j}$ és $RK_{N,j}=K_{N,j}$

$RK_{i,1}=K_{i,1}$ és $RK_{i,M}=K_{i,M}$

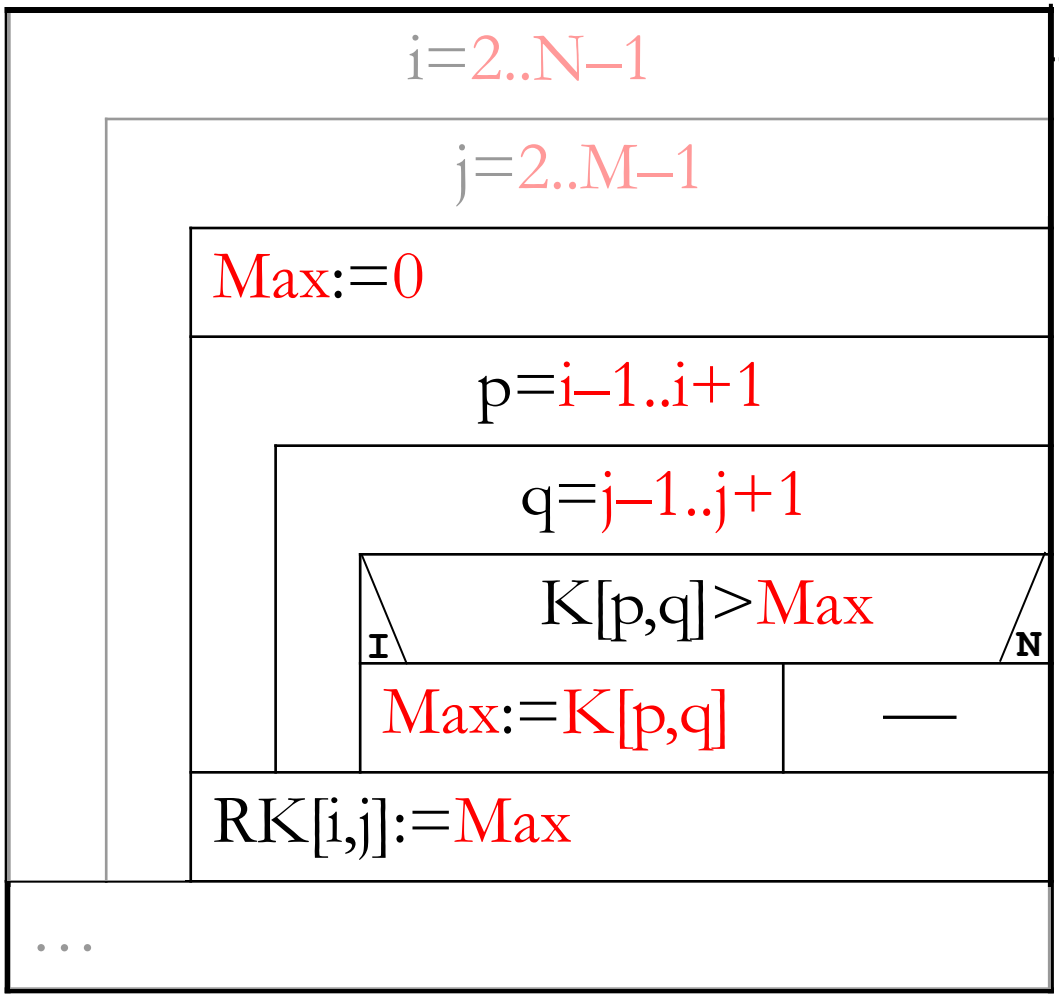
Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $X \in H^N$
- Kimenet: $Max \in \mathbb{N}$
- Előfeltétel: $N>0$
- Utófeltétel: $1 \le Max \le N$ és $\forall i\ (1 \le i \le N): X_{Max} \ge X_i$



Maximumérték-kiválasztás tétel.

Változó
i,j:Egész



Mátrixok

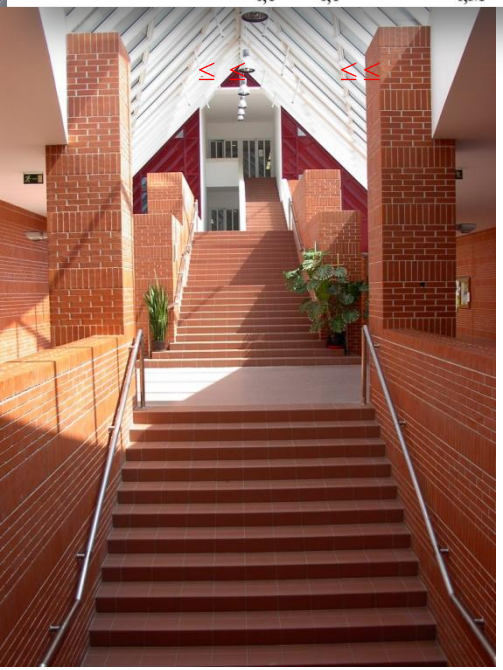
Algoritmus (folytatás):

- Utófeltétel: $\forall i(1 \le i \le N): \forall j(1 \le j \le M):$

$$RK_{i,j} = \max_{p=i-1}^{i+1} \max_{q=j-1}^{j+1} K_{p,q} \quad \text{és}$$

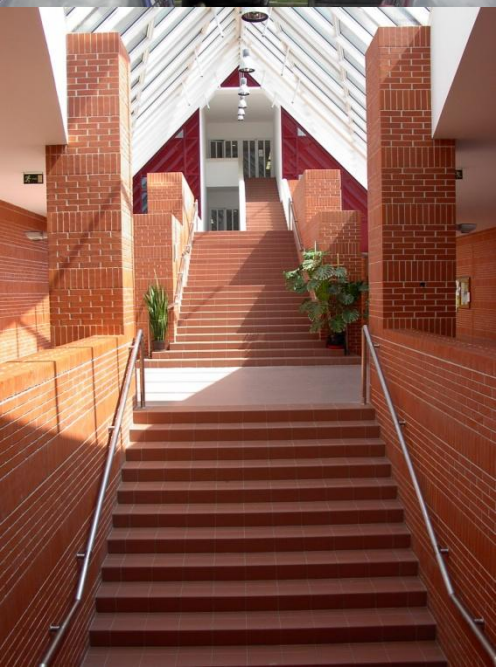
$\forall j(1 \le j \le M)$:
$RK_{1,j}=K_{1,j}$ és $RK_{N,j}=K_{N,j}$

$\forall i(1 \le i \le N)$:
$RK_{i,1}=K_{i,1}$ és $RK_{i,M}=K_{i,M}$



Változó
i,j:**Egész**

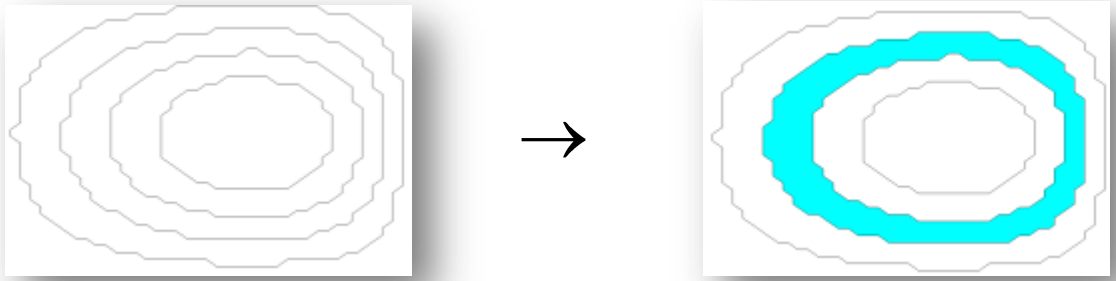
```
...
    j=1..M
        RK[1,j]:=K[1,j]
        RK[N,j]:=K[N,j]
    i=1..N
        RK[i,1]:=K[i,1]
        RK[i,M]:=K[i,M]
```



Mátrixok

Feladat:

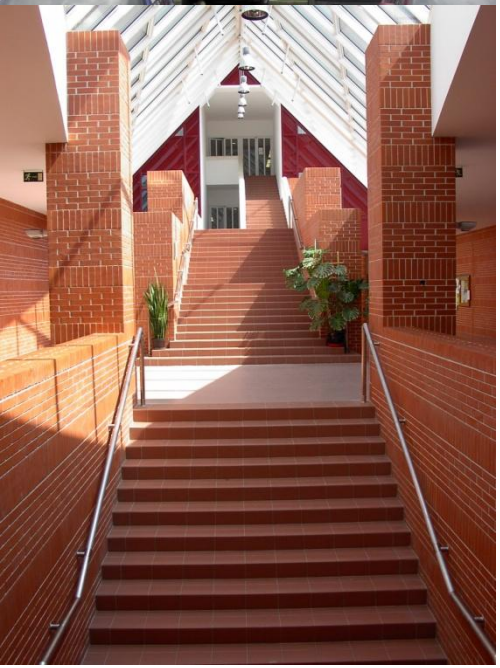
Egy kép egy adott (fehér színű) tartományát egy (A,B) belső pontjából kiindulva fessük be világos-kékre!



Festendők a „**belső pontok**”, ha Belső(i,j)=Igaz.

Ahol Belső:$\mathbb{N}\times\mathbb{N}\rightarrow\mathbb{L}$

Belső(i,j)=(i=A és j=B vagy
Fehér(i,j) és
(Belső(i–1,j) vagy Belső(i+1,j) vagy
Belső(i,j–1) vagy Belső(i,j+1)))



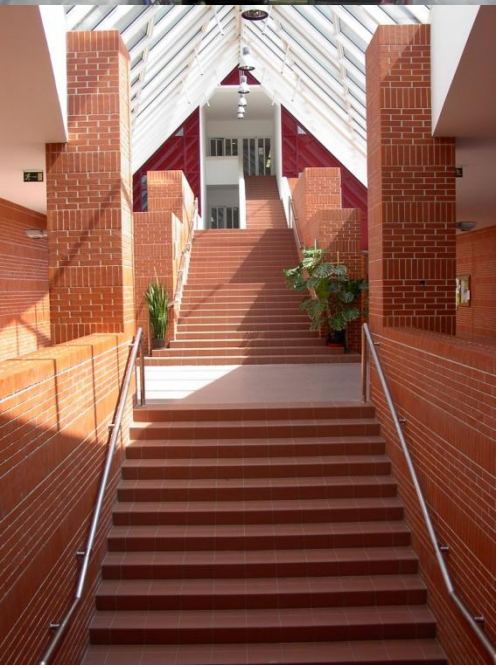
Mátrixok

Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$, $A,B \in \mathbb{N}$
- Kimenet: $KK \in \mathbb{N}^{N \times M}$
- Előfeltétel: –
- Utófeltétel: $\forall i(1 \leq i \leq N)$: $\forall j(1 \leq j \leq M)$:

  Belső(i,j) $\rightarrow KK_{i,j}$=világoskék és

  nem Belső(i,j) $\rightarrow KK_{i,j}=K_{i,j}$



Mátrixok

Specifikáció:

- Bemenet: $N,M \in \mathbb{N}$, $K \in \mathbb{N}^{N \times M}$, $A,B \in \mathbb{N}$
- Kimenet: $KK \in \mathbb{N}^{N \times M}$
- Előfeltétel: –
- Utófeltétel: $\forall i(1 \leq i \leq N)$: $\forall j(1 \leq j \leq M)$:
  Belső(i,j) $\rightarrow KK_{i,j}$=világoskék és
  nem Belső(i,j) $\rightarrow KK_{i,j}=K_{i,j}$

Algoritmus:

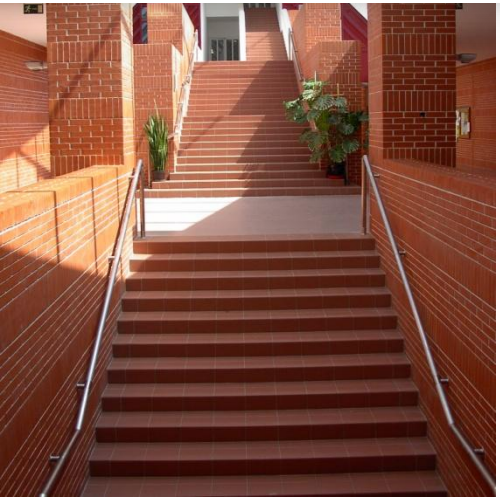
KK:=K
Festés(A,B)

Mátrixok

Algoritmus:

- Utófeltétel: $\forall i(1 \le i \le N)$: $\forall j(1 \le j \le M)$:
  Belső(i,j) → $KK_{i,j}$=világoskék és
  nem Belső(i,j) → $KK_{i,j}=K_{i,j}$

- Definíció:
  Belső(i,j)=(i=A és j=B vagy
  Fehér(i,j) és
  (Belső(i–1,j) vagy Belső(i+1,j) vagy
  Belső(i,j–1) vagy Belső(i,j+1)))



Festés(i,j:**Egész**)

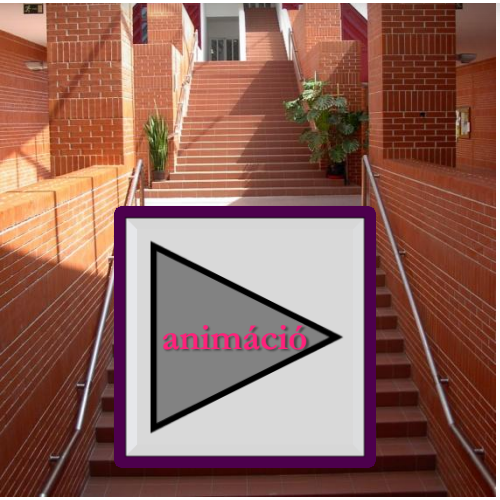
KK[i,j]:=világoskék	
I: KK[i–1,j]=fehér — N	
Festés(i–1,j)	—
I: KK[i+1,j]=fehér — N	
Festés(i+1,j)	—
I: KK[i,j–1]=fehér — N	
Festés(i,j–1)	—
I: KK[i,j+1]=fehér — N	
Festés(i,j+1)	—

Mátrixok

Algoritmus:

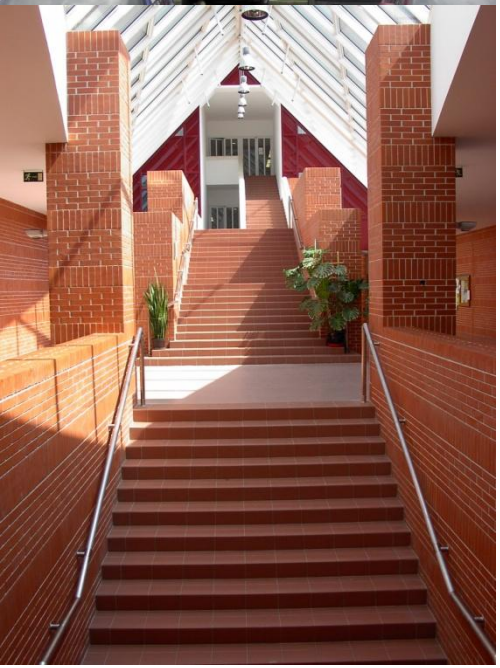
- Utófeltétel: ∀i(1≤i≤N): ∀j(1≤j≤M):
  Belső(i,j) → $KK_{i,j}$=világoskék és
  nem Belső(i,j) → $KK_{i,j}=K_{i,j}$

- Definíció:
  Belső(i,j)=(i=A és j=B vagy
  Fehér(i,j) és
  (Belső(i–1,j) vagy Belső(i+1,j) vagy
  Belső(i,j–1) vagy Belső(i,j+1)))



Festés(i,j:**Egész**)

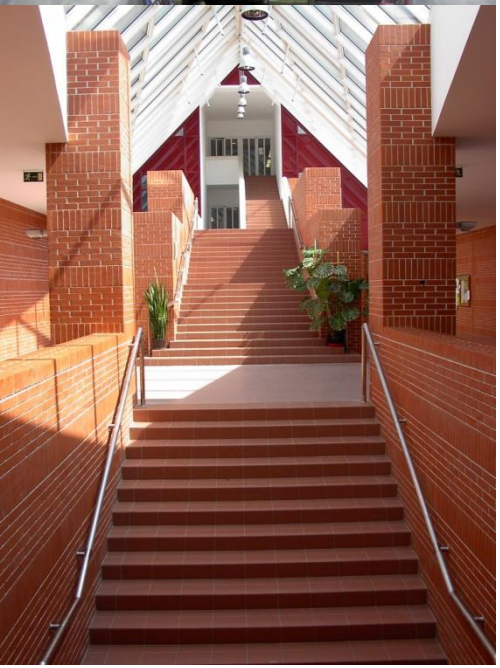
KK[i,j]:=világoskék		
I	KK[i–1,j]=fehér és i>1	N
Festés(i–1,j)	—	
I	KK[i+1,j]=fehér és i<N	N
Festés(i+1,j)	—	
I	KK[i,j–1]=fehér és j>1	N
Festés(i,j–1)	—	
I	KK[i,j+1]=fehér és j<M	N
Festés(i,j+1)	—	



Rekordok vektora

Feladat:

Egy adott napon N-szer volt földrengés. Ismerjük az egyes rengések időpontját. Mondjuk meg, hogy hány másodpercenként volt földrengés!



Rekordok vektora

Feladat:

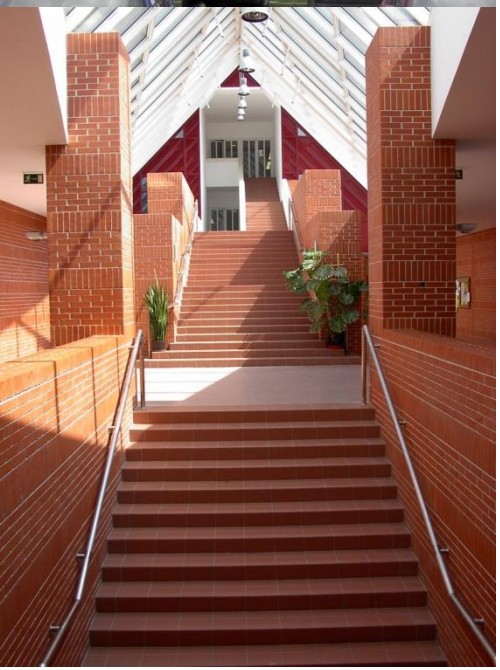
Egy adott napon N-szer volt földrengés. Ismerjük az egyes rengések időpontját. Mondjuk meg, hogy hány másodpercenként volt földrengés!

Megoldás felé:

- Definiálni kellene, mi az idő!
  Az időt megadhatjuk az (óra, perc, másodperc) hármassal, azaz az idő:
  $$\text{Idő} = \text{Ó} \times \text{P} \times \text{Mp}, \quad \text{Ó}, \text{P}, \text{Mp} = \mathbb{N}$$
- Algoritmikus sablon: **Másolás** tétel!

Feladat:

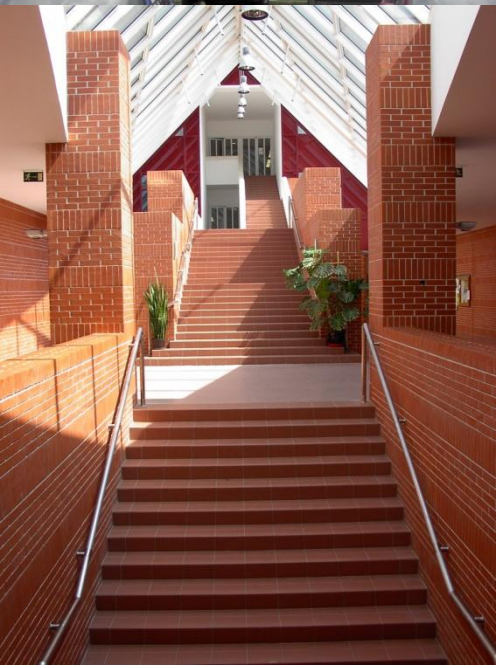
Egy adott napon N-szer volt földrengés. Ismerjük az egyes rengések időpontját. Mondjuk meg, hogy hány másodpercenként volt földrengés!



Rekordok vektora

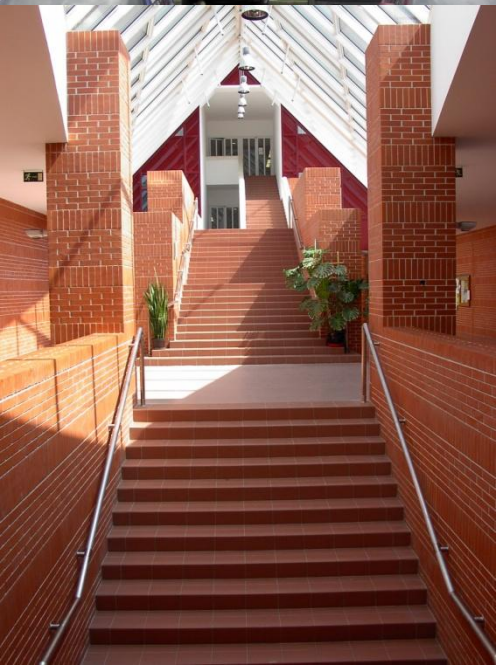
Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $R \in \text{Idő}^N$
  $\text{Idő} = Ó \times P \times Mp$, $Ó, P, Mp = \mathbb{N}$
- Kimenet: $T \in \mathbb{N}^{N-1}$
- Előfeltétel: $\forall i (1 \le i \le N)$: $0 \le R_i.ó \le 23$ és
  $0 \le R_i.p \le 59$ és $0 \le R_i.mp \le 59$ és
  $\forall i (1 \le i < N)$: $R_i < R_{i+1}$
- Utófeltétel: $\forall i (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció: $- : \text{Idő} \times \text{Idő} \to \mathbb{N}$
  $i1 - i2 := \ldots ??? \ldots$
  $< : \text{Idő} \times \text{Idő} \to \mathbb{L}$
  $i1 < i2 := \ldots ??? \ldots$



Rekordok vektora

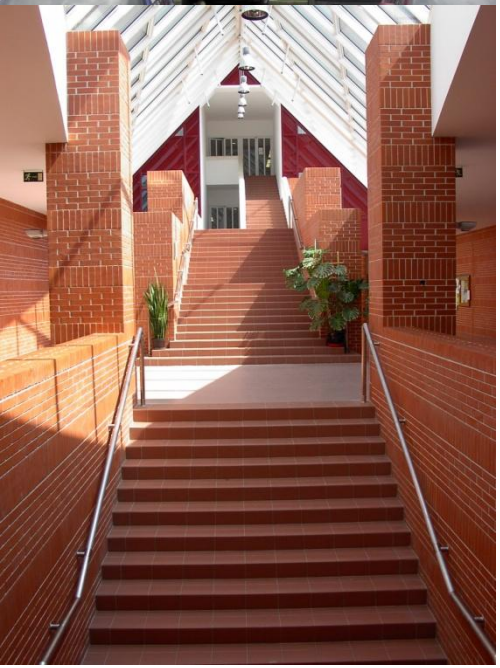
Idők különbsége



Rekordok vektora

Idők különbsége

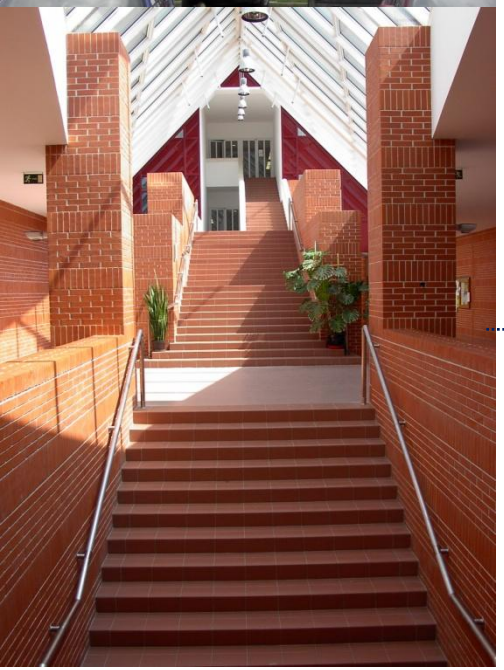
1. megoldási ötlet:
   Felfoghatjuk úgy, mint két háromjegyű szám különbsége, ahol a három jegy nem azonos alapú. (Vegyes alapú számrendszer.)
   Majd másodpercekké konvertáljuk.



Rekordok vektora

Idők különbsége

1. megoldási ötlet:
   Felfoghatjuk úgy, mint két háromjegyű szám különbsége, ahol a három jegy nem azonos alapú. (Vegyes alapú számrendszer.)
   Majd másodpercekké konvertáljuk.
2. megoldási ötlet:
   Kifejezzük az időket másodpercben, így már két egész szám különbségét kell kiszámolni.



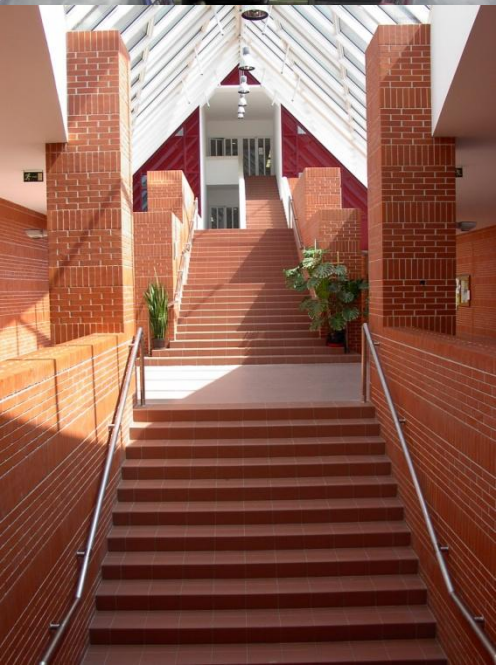
Rekordok vektora

Idők különbsége

1. megoldási ötlet:
   Felfoghatjuk úgy, mint két háromjegyű szám különbsége, ahol a három jegy nem azonos alapú. (Vegyes alapú számrendszer.)
   Majd másodpercekké konvertáljuk.
2. megoldási ötlet:
   Kifejezzük az időket másodpercben, így már két egész szám különbségét kell kiszámolni.

másodpercben(i):=i.ó*3600+i.p*60+i.mp

Meggondolandó, h. mekkora egész szám kell hozzá? (24*3600=86 400) Milyen típusú lehet? (>2 byte)



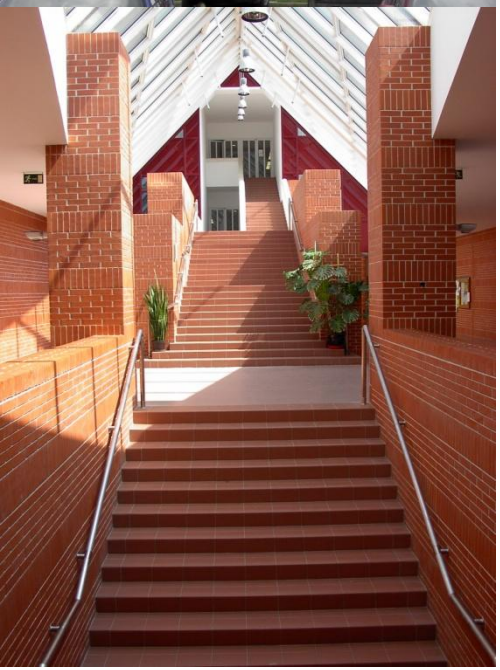
Rekordok vektora

Specifikáció:

- Bemenet: $N \in \mathbb{N}$, $R \in \text{Idő}^N$
  $\text{Idő} = Ó \times P \times Mp$, $Ó, P, Mp = \mathbb{N}$
- Kimenet: $T \in \mathbb{N}^{N-1}$
- Előfeltétel: $\forall i (1 \le i \le N)$: $0 \le R_i.ó \le 23$ és
  $0 \le R_i.p \le 59$ és $0 \le R_i.mp \le 59$ és
  $\forall i (1 \le i < N)$: $R_i < R_{i+1}$
- Utófeltétel: $\forall i (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció:
  $- : \text{Idő} \times \text{Idő} \to \mathbb{N}$
  $i1 - i2 := i1.ó*3600+i1.p*60+i1.mp - (i2.ó*3600+i2.p*60+i2.mp)$

Rekordok vektora

- Utófeltétel: $\forall i\ (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció:
  - $-$: Idő×Idő→N
  - i1 − i2 := i1.ó*3600+i1.p*60+i1.mp − (i2.ó*3600+i2.p*60+i2.mp)



Algoritmus$_1$:

```
Változó
  i:Egész
  S:Tömb[…]
```

```
i=1..N
    S[i]:=R[i].ó*3600+
          R[i].p*60+R[i].mp
i=1..N–1
    T[i]:=S[i+1]–S[i]
```

Megjegyzések:

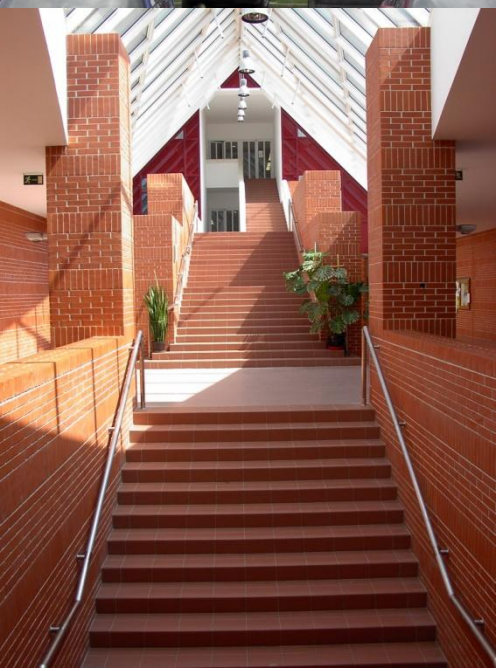
1. Egy S segédtömböt használunk.
2. A TIdők közötti „–” operátor az S-en keresztül, közvetve kerül az algoritmusba.

Rekordok vektora

Algoritmus$_2$:

- Utófeltétel: $\forall i\ (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció:

másodpercben(i):=i.o*3600+i.p*60+i.mp



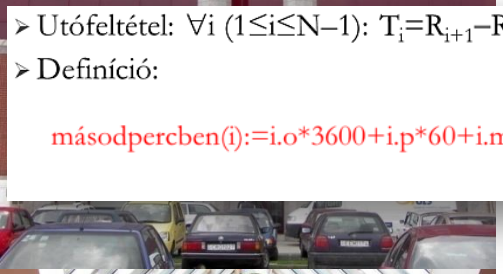
Változó
i:**Egész**
S:Tömb[…]

```
i=1..N
    S[i]:=másodpercben(R[i])
i=1..N–1
    T[i]:=S[i+1]–S[i]
```

Megjegyzések:

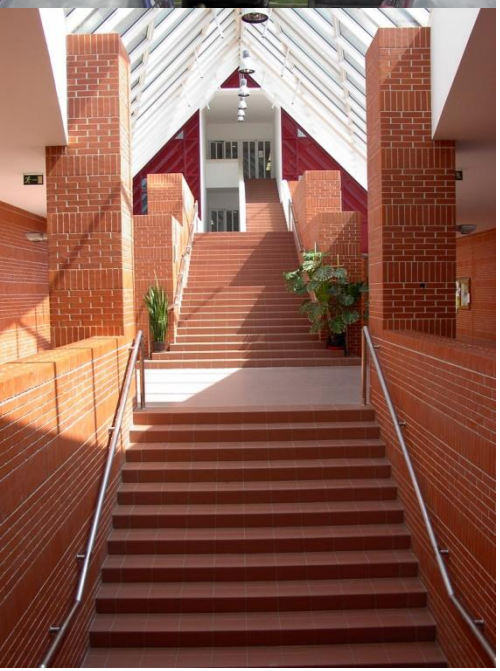
1. A másodpercben fv. megvalósítandó!
2. Ha a különbség (óra, perc, másodperc)-ben kell, akkor T[i]-ből vissza kell alakítani!
   Újabb művelet.

Rekordok vektora



- Utófeltétel: $\forall i\ (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció:

másodpercben(i):=i.o*3600+i.p*60+i.mp



Algoritmus$_3$:

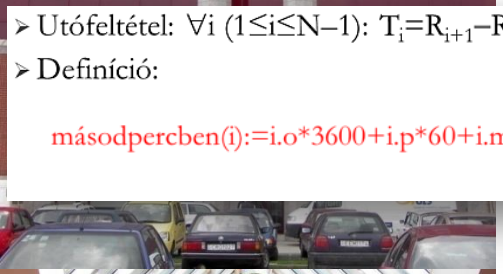
Változó
i:Egész

```
i=1..N–1
    T[i]:=másodpercben(R[i+1])–
          másodpercben(R[i])
```

Megjegyzés:

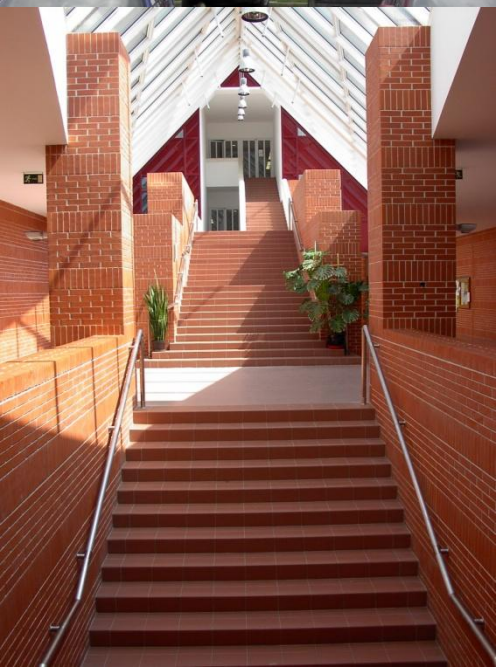
A másodpercben fv. segítségével (sőt anélkül is) megspórolható az S segédtömb; és így az előkészítő ciklus… de cserében majdnem minden R[i]-t kétszer számítunk át másodpercekre.

Rekordok vektora



- Utófeltétel: $\forall i\ (1 \le i \le N-1)$: $T_i = R_{i+1} - R_i$
- Definíció:

másodpercben(i):=i.o*3600+i.p*60+i.mp



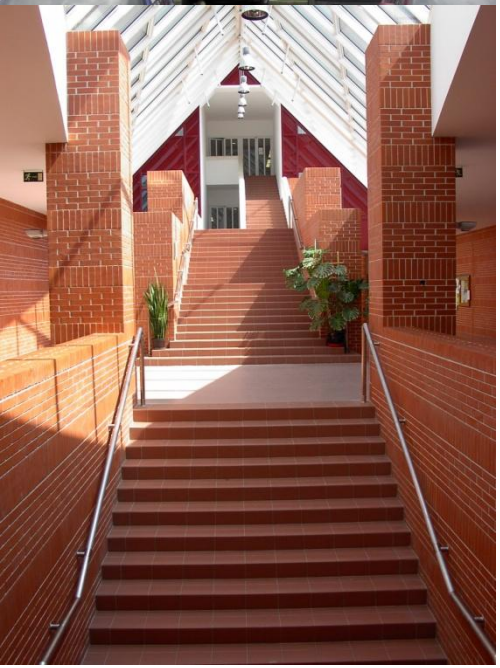
Algoritmus$_4$:

Változó
i:Egész

```
i=1..N–1
    T[i]:=R[i+1]–R[i]
```

Megjegyzés:

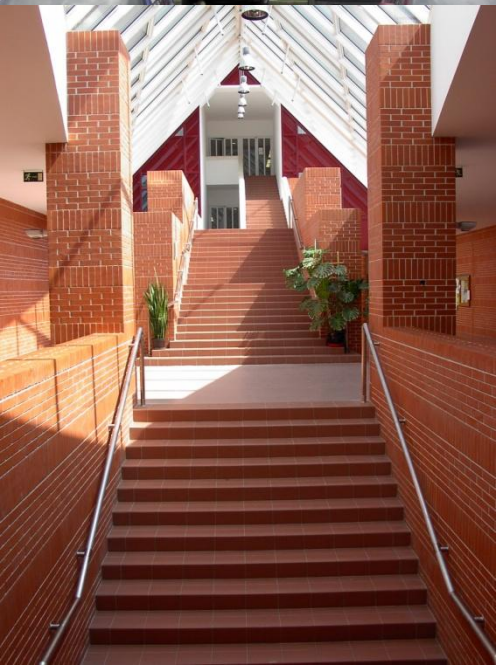
A – operátort definiálni kell, amelyben a másodpercekben fv. (v. annak törzse) felhasználható!



Vektorok rekordja

Feladat:

Ismerjük egy ember összes telefonszámát és e-levél címét. Egy adott telefonszámról és e-levél címről el kell döntenünk, hogy lehet-e az adott emberé!



Vektorok rekordja

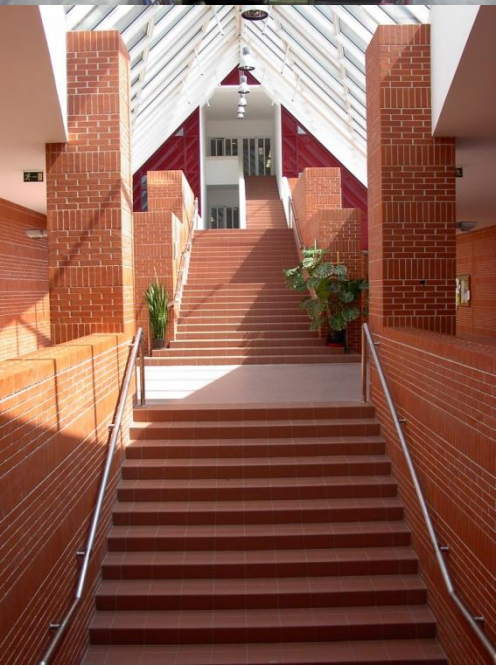
Feladat:

Ismerjük egy ember összes telefonszámát és e-levél címét. Egy adott telefonszámról és e-levél címről el kell döntenünk, hogy lehet-e az adott emberé!

Kérdések/válaszok:

- Hogyan ábrázoljuk a specifikációban?

$$\text{Ember}=Dbt\times Dbe\times Telefon\times Elevél,$$
$$Dbt,Dbe=\mathbb{N},\ Telefon=\mathbb{S}^{Dbt},\ Elevél=\mathbb{S}^{Dbe}$$



Vektorok rekordja

Feladat:

Ismerjük egy ember összes telefonszámát és e-levél címét. Egy adott telefonszámról és e-levél címről el kell döntenünk, hogy lehet-e az adott emberé!

Kérdések/válaszok:

- Hogyan ábrázoljuk a specifikációban?

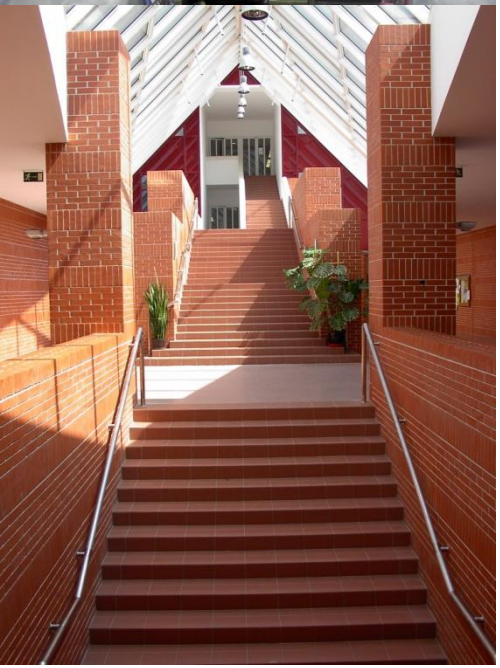
  $Ember = Dbt \times Dbe \times Telefon \times Elevél$,

  $Dbt, Dbe = \mathbb{N}$, $Telefon = \mathbb{S}^{Dbt}$, $Elevél = \mathbb{S}^{Dbe}$

- Hogyan ábrázoljuk az algoritmusban?

```
TEmber=Rekord(
        dbt,dbe:Egész,
        telefon:Tömb[1..MaxT:Szöveg],
        elevél: Tömb[1..MaxE:Szöveg] )
```

Feladat:
Ismerjük egy ember összes telefonszámát és e-levél címét. Egy adott telefonszámról és e-levél címről el kell döntenünk, hogy lehet-e az adott emberé!



Vektorok rekordja

Specifikáció:

- Bemenet: $X \in \text{Ember}$, $Tel, Elev \in \mathbf{S}$,
  $\text{Ember} = Dbt \times Dbe \times Telefon \times Elevél$,
  $Dbt, Dbe = \mathbb{N}$, $Telefon = \mathbb{S}^{Dbt}$, $Elevél = \mathbb{S}^{Dbe}$
  Kimenet: $Lehet \in \mathbf{L}$
- Előfeltétel: …
- Utófeltétel: $Lehet = Tel \in X.telefon$ és
  $Elev \in X.elevél$

Feladat:
Ismerjük egy ember összes telefonszámát és e-levél címét. Egy adott telefonszámról és e-levél címről el kell döntenünk, hogy lehet-e az adott emberé!

Specifikáció:
- Bemenet: N∈ℕ,
  X∈H^N,
  T:H→L
- Kimenet: Van∈L
- Előfeltétel: –
- Utófeltétel: Van=∃i(1≤i≤N): T(X_i)

Vektorok rekordja

Specifikáció:

- Bemenet: X∈Ember, Tel, Elev∈**S**,
  Ember=Dbt×Dbe×Telefon×Elevél,
  Dbt,Dbe=ℕ, Telefon=$\mathbb{S}^{Dbt}$, Elevél=$\mathbb{S}^{Dbe}$
  Kimenet: Lehet∈**L**
- Előfeltétel: …
- Utófeltétel:Lehet=Tel∈X.telefon és
  Elev∈X.elevél

Megjegyzés:

s∈S ↔ s=S_1 vagy s=S_2 … →

→ Algoritmikus sablon: **eldöntés** tétel

Vektorok rekordja

Algoritmus:

Specifikáció:

- Bemenet: X∈Ember, Tel, Elev∈**S**,
  Ember=Dbt×Dbe×Telefon×Elevél,
  Dbt,Dbe=N, Telefon=S^{Dbt}, Elevél=S^{Dbe}
  Kimenet: Lehet∈**L**
- Előfeltétel: …
- Utófeltétel: Lehet=Tel∈X.telefon és
  Elev∈X.elevél

Változó
i,j:**Egész**

i:=1

i≤X.dbt és X.telefon[i]≠Tel

i:=i+1

i≤X.dbt

I N

j:=1

j≤X.dbe és X.elevél[j]≠Elev

j:=j+1

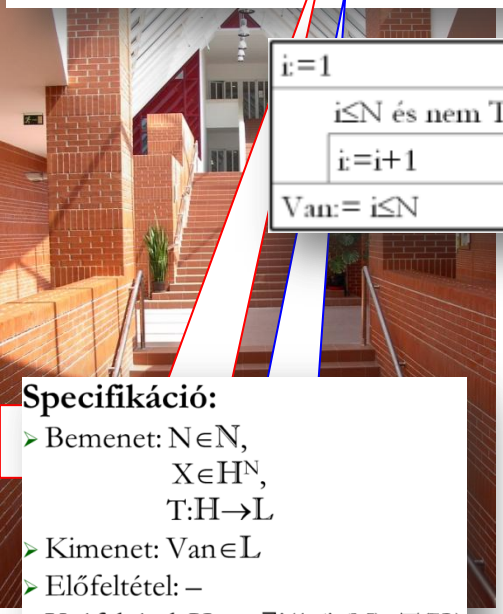
—

Lehet:= i≤X.dbt és j≤X.dbe

Vektorok rekordja

Specifikáció:

- Bemenet: X∈Ember, Tel, Elev∈**S**,
  Ember=Dbt×Dbe×Telefon×Elevél,
  Dbt,Dbe=N, Telefon=S^{Dbt}, Elevél=S^{Dbe}
  Kimenet: Lehet∈**L**
- Előfeltétel: …
- Utófeltétel: Lehet=Tel∈X.telefon és
  Elev∈X.elevél



i:=1
i≤N és nem T(X[i])
i:=i+1
Van:= i≤N

Specifikáció:

- Bemenet: N∈N,
  X∈H^N,
  T:H→L
- Kimenet: Van∈L
- Előfeltétel: –
- Utófeltétel: Van=∃i(1≤i≤N): T(X_i)

Algoritmus:

Változó
i,j:**Egész**

i:=1

i≤X.dbt és X.telefon[i]≠Tel

i:=i+1

i≤X.dbt

I N

j:=1

j≤X.dbe és X.elevél[j]≠Elev

j:=j+1

—

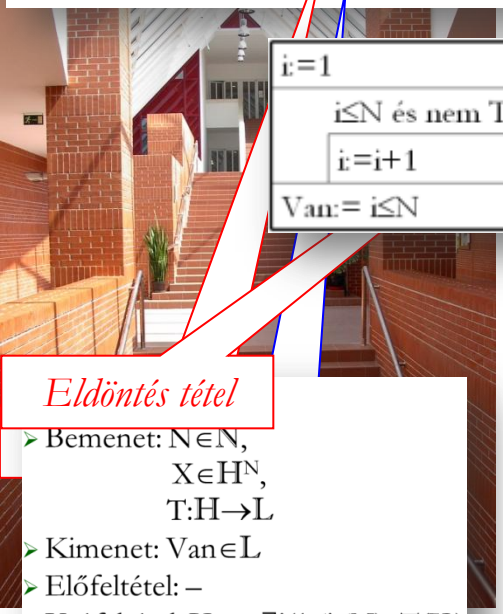
Lehet:= i≤X.dbt és j≤X.dbe

Vektorok rekordja

Algoritmus:

Specifikáció:

- Bemenet: X∈Ember, Tel, Elev∈**S**,
  Ember=Dbt×Dbe×Telefon×Elevél,
  Dbt,Dbe=N, Telefon=S^{Dbt}, Elevél=S^{Dbe}
  Kimenet: Lehet∈**L**
- Előfeltétel: …
- Utófeltétel: Lehet=Tel∈X.telefon és
  Elev∈X.elevél



i:=1
i≤N és nem T(X[i])
i:=i+1
Van:= i≤N

Eldöntés tétel

- Bemenet: N∈N,
  X∈H^N,
  T:H→L
- Kimenet: Van∈L
- Előfeltétel: –
- Utófeltétel: Van=∃i(1≤i≤N): T(X_i)

Változó
i,j:**Egész**

i:=1

i≤X.dbt és X.telefon[i]≠Tel

i:=i+1

i≤X.dbt

I N

j:=1

j≤X.dbe és X.elevél[j]≠Elev

j:=j+1

—

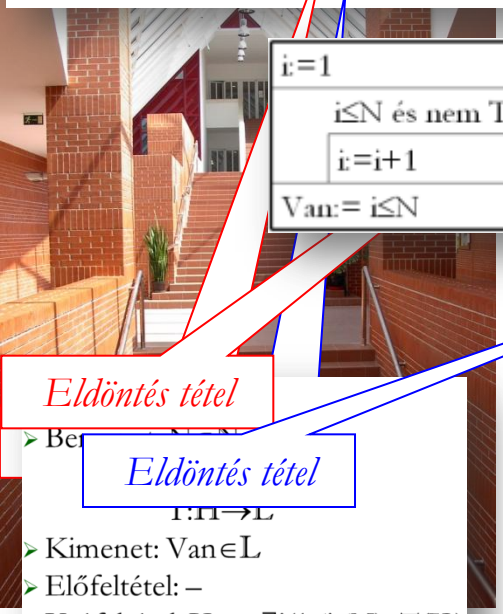
Lehet:= i≤X.dbt és j≤X.dbe

Vektorok rekordja

Algoritmus:

Specifikáció:

- Bemenet: X∈Ember, Tel, Elev∈**S**,
  Ember=Dbt×Dbe×Telefon×Elevél,
  Dbt,Dbe=N, Telefon=S^{Dbt}, Elevél=S^{Dbe}
  Kimenet: Lehet∈**L**
- Előfeltétel: …
- Utófeltétel: Lehet=Tel∈X.telefon és
  Elev∈X.elevél



i:=1
i≤N és nem T(X[i])
i:=i+1
Van:= i≤N

Eldöntés tétel

- Be
- Kimenet: Van∈L
- Előfeltétel: –
- Utófeltétel: Van=∃i(1≤i≤N): T(X$_i$)

Eldöntés tétel

Változó
i,j:Egész

i:=1
i≤X.dbt és X.telefon[i]≠Tel
i:=i+1

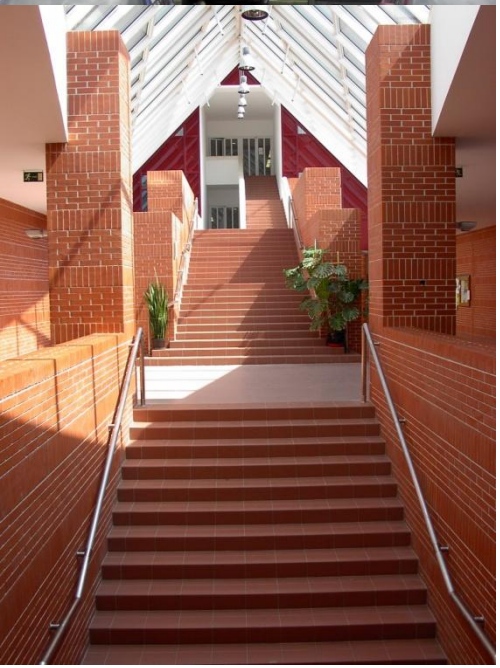
i≤X.dbt

I N

j:=1
j≤X.dbe és X.elevél[j]≠Elev
j:=j+1

—

Lehet:= i≤X.dbt és j≤X.dbe



Vektor-rekord

További példák:

KÉP[i,j].piros – az (i,j) képpont RGB kódjának piros része

EMBER[j].telefon[i].körzetszám – a j-edik ember i-edik telefonszámának a körzetszáma

T.lap[i].él[j].pont[1].x –a T test i-edik lapja j-edik éle 1. végpontjának x-koordinátája

Programozási alapismeretek
8. előadás vége