|  |  |  |
| --- | --- | --- |
| **Ács Máté Olivér**  GI1DO1  acsmateoli@inf.elte.hu  18.csoport | **1. beadandó/1.feladat** | 2020. március 14.[[1]](#footnote-1) |

# Feladat

Valósítsa meg az egész számokat tartalmazó „sakktábla” mátrixtípust. Ezen m×n-es mátrixok

soraiban biztosan nulla értékű minden második elem. A „nem-nulla” értékek sakktábla-szerűen

helyezkednek el az [1,1], [1,3], ... , [2,2], [2,4], ... indexű helyeken. A típus reprezentációjában

csak ezeket a „nem-nulla” értékű elemeket kell eltárolnunk. (Az [1,2], [1,4], ... , [2,1], [2,3], ...

indexű helyeken levő biztosan nulla értékű elemeket nem tároljuk.) Implementálja önálló

metódusként a mátrix i-edik sorának j-edik elemét visszaadó műveletet, valamint az összeadás

és szorzás műveleteket, továbbá a mátrix m×n alakban történő kiírását!

# Sakktábla mátrix típus

## Típusérték-halmaz

Olyan *n*×*n*-es (*n*∈ℕ) mátrixok, amelynek elemei egész számok, és csak a minden második elem tartalmazhat nullától különböző elemeket. (Az *n*∈ℕ ennek a típusnak egy paramétere, amely a típusérték-halmaz mátrixainak méreteit határozzák meg.)

Formálisan: *Sakk*(*n*) = { *a*∈ ℤn×n | ∀*i,j*∈ [*1*..*n*]: 2|i ∧ ⌈2|j ∨ ⌈2|i ∧ 2|j → *a*[*i,j*]=*0* }

## Típus-műveletek

### 1. Lekérdezés

A mátrix *i*-edik sorának *j-*edik pozícióján (*i,j*∈ [*1*..*n*]) álló érték kiolvasása: *e*:=*a*[*i,j*].

Formálisan: *A* = ( *a* : *Sakk*(*n*), *i* : ℤ, *j* : ℤ, *e* : ℤ )

*Ef* = ( *a*=*a’* ∧ *i*=*i’* ∧ *j*=*j’* ∧ *i,j*∈ [*1*..*n*])

*Uf* = ( *Ef* ∧ *e*=*a*[*i,j*] )

### 2. Összeadás

Két mátrix összeadása: *c:=a+b*. Az összeadásban szereplő mátrixok azonos méretűek.

Formálisan: *A* = ( *a* : *Sakk*(*n*), *b* : *Sakk*(*n*), *c* : *Sakk*(*n*) )

*Ef* = ( *a*=*a’* ∧ *b*=*b’*)

*Uf* = ( *Ef* ∧ ∀*i,j*∈ [*1*..*n*]: *c*[*i*,*j*]= *a*[*i*,*j*] + *b*[*i*,*j*] )

|  |  |
| --- | --- |
| *3. Szorzás*  Két mátrix szorzása: *c:=a\*b*. Az szorzásban szereplő mátrixok azonos méretűek.    Formálisan: *A* = ( *a* : *Sakk*(*n*), *b* : *Sakk*(n) *c* : *Sakk*(*n*) )  *Ef* = ( *a*=*a’* ∧ *b*=*b’*)  *Uf* = ( *Ef* ∧∀*i,j*∈ [*1*..*z*]: *c*[*i*,*j*]=∑*k=1..n**a*[*i*,*k*] \* *b*[*k*,*j*]) |  |
|  |  |

## Reprezentáció

Egy nxn-es mátrixnak csak minden második elemét kell ábrázolni, azaz egy n\*n darab elemet tartalmazó mátrix helyett elég csak minden második elemet tárolni egy 0-tól indexelt egydimenziós tömbben (vec).

|  |  |  |  |
| --- | --- | --- | --- |
| a11 | 0 | a13 | …0 |
| 0 | a22 | 0 | …a2n |
| a31 | 0 | a33 | …0 |
| 0 | an2 | 0 | …ann |

a=

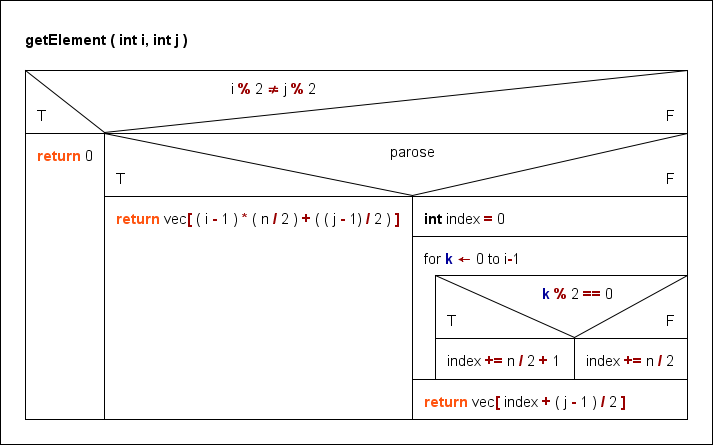
↔ v= <a11, a13, a22, a24, a31, a33, a42, ann>

## Implementáció

Semelyik implementációm nem tartalmaz programozási tételt.

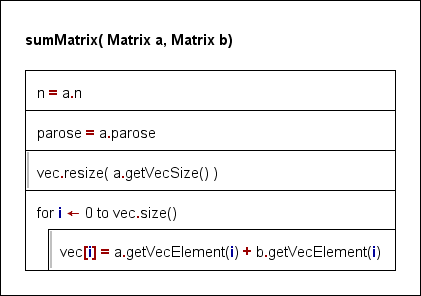
### Lekérdezés

Egy mátrixra, amikor ráhívjuk ezt a függvényt, ellenőrzi először, hogy a megadott (i,j) indexek nem kisebbek nullánál vagy esetleg nagyobbak, mint maga a mátrix, ezekben az esetekben kivételt dobunk, amit később a program elkap. Amennyiben az indexek megfelelőek, a következő program fut le:



### 2. Összeadás

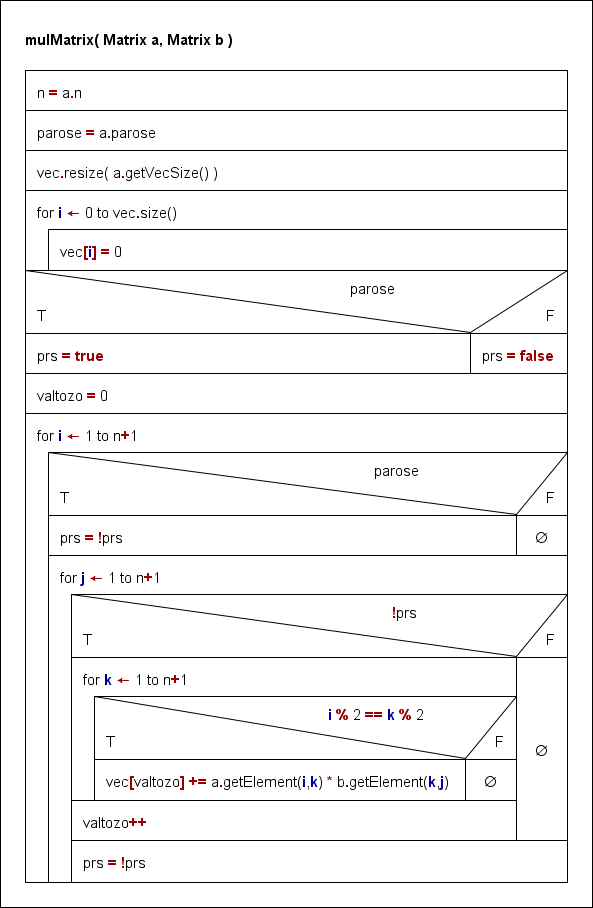
Egy mátrixra, amiben az eredmény mátrixot fogjuk tárolni hívjuk meg ezt a függvényt, amely ellenőrzi, hogy ha a két paraméterként kapott mátrixok mérete nem egyezik meg, akkor kivételt dobjunk, amit majd a program később elkap. Amennyiben a méretek megfelelőek, a következő program fut le:



(A getVecSize() függvény az adott vektor hosszát adja vissza, míg a getVecElement(int i) az adott vektor i-edik elemét adja vissza)

### 3. Szorzás

Egy mátrixra, amiben az eredmény mátrixot fogjuk tárolni hívjuk meg ezt a függvényt, amely ellenőrzi, hogy ha a két paraméterként kapott mátrixok mérete nem egyezik meg, akkor kivételt dobjunk, amit majd a program később elkap. Amennyiben a méretek megfelelőek, a következő program fut le:



1. Végleges, javított verzió elkészülése: március 18 [↑](#footnote-ref-1)