**Feladat**

*Valósítsa meg az egész számokat tartalmazó „N-mátrixtípust”. Ezek négyzetes mátrixok, amelyek csak a szélső oszlopokban, illetve a főátlójában tartalmazhatnak nullától különböző elemeket. Ilyenkor elegendő csak a nem 0 elemeket eltárolni egy sorozatban. Implementálja önálló metódusként a mátrix iedik sorának j-edik elemét visszaadó műveletet, valamint hatékony összeadás és szorzás műveleteket, továbbá a mátrix (négyzetes alakú) kiírását, és végül a másoló konstruktort és az értékadás operátort!*

**N-mátrixtípus**

A feladat lényege egy felhasználói típusnak az N-mátrixtípus megvalósítása.

**1.Típusérték-halmaz**

Olyan egész számokat számokat tartalmazó n×n-es (n∈N) négyzetes mátrixokkal akarunk dolgozni, amelyek csak az első és utolsó oszlopukban, illetve a főátlóban tartalmaznak 0-tól különböző elemet. Az n∈N ennek a típusnak egy parameter, amely a matrix méretét határozza meg.

Formálisan: Nmatrix(n)= {A∈Zn×n | ∀ i,j ∈[1…n]:j≠1 ∧ j≠n ∧ i≠j →A[i,j]=0}

**2. Típus-műveletek**

1.*Lekérdezés*

A=(A:*Nmatrix*,i: N,j:N,e:Z)

Ef=(A=A’ ∧ i=i’ ∧ j=j’)

Uf=(Ef ∧ e:=A[i,j])

2.*Összeadás*

A=(A:*Nmatrix*,B:*Nmatrix*,C:*Nmatrix*)

Ef=(A=A’ ∧ B=B’)

Uf=(Ef ∧ ∀i,j∈[1…n] : C[i,j]=A[i,j]+B[i,j])

3.*Szorzás*

A=(A:*Nmatrix*, B:*Nmatrix*, C:*Nmatrix*)

Ef=(A=A’ ∧ B=B’)

Uf=(Ef ∧ ∀i,j∈[1…n]: C[i,j]=)

Az N-mátrixtípusok esetében a fenti egyenlet sokkal hatékonyabban megfogalmazható a következőképpen:

C[i,j]=

**3.Reprezentáció**

Az n×n-es matrix első és utolsó oszlopában, illetve a főátlójában lévő elemeket kell ábrázolni:

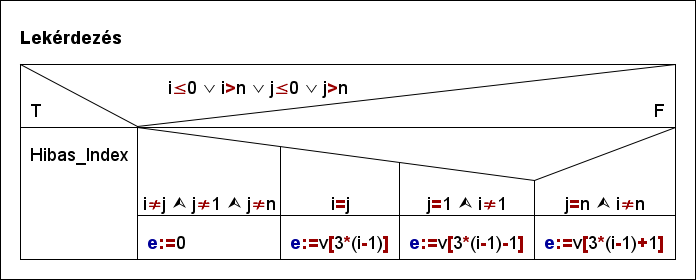
→ a11, a1n,a21,a22,a2n,..,an1,ann

Ehhez egy 0-tól ( 3\*n-2)-ig indexelt egydimenziós tömbre van szükségünk (v). Ennek a segítségével az N-típusú matrix bármelyik elemét meghatározhatjuk.

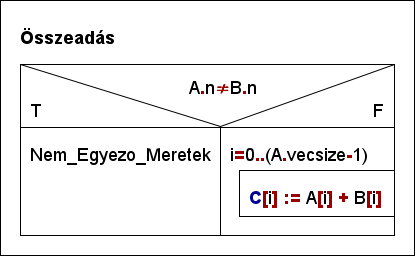
**4.Implementáció**

1.*Lekérdezés*

A *v* tömbbel ábrázolt n×n-es N-típusú matrix i-edik sorának j-edik *e* elemét a következőképpen kapjuk meg:

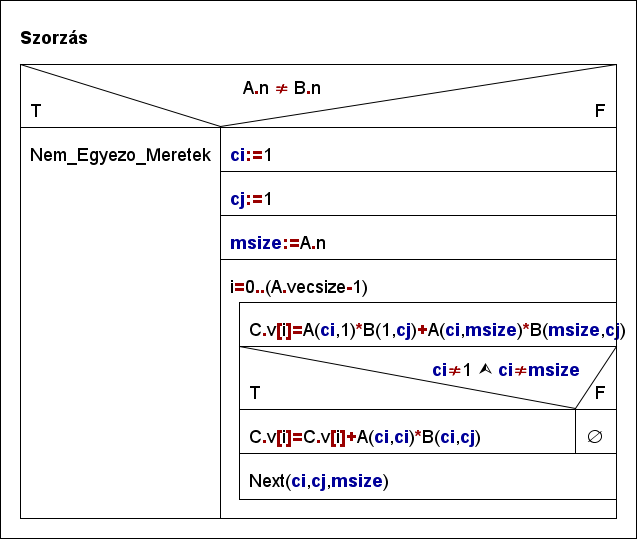


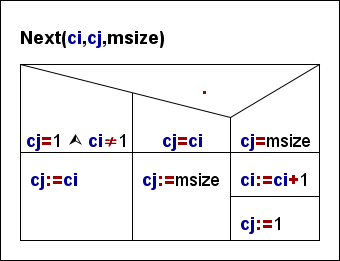
2.*Összeadás*

Az N-típusú A és B mátrix C összegét a következő algoritmussal kapjuk meg, ahol a *vecsize* változó a *v* tömb méretét tárolja (vecsize=3\*n-2).

3.*Szorzás*

Az N-típusú A és B mátrix C szorzatát a következő algoritmussal kapjuk meg, ahol a *vecsize* változó a *v* tömb méretét tárolja (vecsize=3\*n-2).

A *Next(ci,cj,msize)* függvény segít meghatározni, hogy az algoritmus a C mátrix melyik sorának (ci) hanyadik elemét (cj) számolja ki és a következőképpen néz ki:



**Osztály**

Az N-mátrixtípust egy osztály segítségével van megvalósítva. A konstruktoron keresztül állítható be a mátrix mérete. A műveleteknél és az értékadás operátornál ellenőrízve van, hogy a két mátrix mérete megegyezik. Amennyiben nem a program kivételt dob. Az elemek dinamikus tömbben vannak tárolva. A konstruktor végzi a dinamikus tömb lefoglalását, a destruktor pedig a felszabadítást.

Egy mátrix i-edik sorának j-edik elemét a () operátorral lehet kiolvasni, az összeadás és szorzás műveletek külső barát függvényként felüldefiniált változataival vannak megvalósítva.

Az osztály a mátrix kiírását és beolvasását végző metódusokkal van kiegészítve, amelyek a kiíró és beolvasó operátorok külső barát függvényként felüldefiniált változataival vannak megvalósítva.

A hibakezelésre három kivétel van definiálva: Hibas\_Index, Hibas\_Meret, Nem\_Egyezo\_Meretek. A Hibas\_Index a hibásan megadott index esetén, a Hibas\_Meret a konstruktornál 0 vagy annál kisebb méret esetén, a Nem\_Egyezo\_Meretek pedig a összeadásnál, szorzásnál és értékadás operátornál két különböző méretű mátrix esetén váltódik ki.

**Tesztelési terv**

A teszteléshez létre van hozva egy Menu osztály, amely lehetőséget nyújt a felhasználónak egy 3×3-as mátrix beolvasására és kiiratására, illetve két 3×3-as mátrix beolvasására, majd ezek összeadására vagy összeszorzására.

További tesztelés céljából három input fájl van létrehozva, mind egy vagy több különböző méretű N-típusú mátrix értékeivel. Ezek segítvégével az *Nmatrix* összes metódusa tesztelve van a következőképpen:

1. Létrehozás és lekérdezés

Ebben a tesztesetben 3 különböző méretű mátrixot hozunk létre, ezeket beolvassuk az *input1.txt* fájlból, majd leellenőrízzük, hogy a beolvasás helyesen történt és megfelelő indexekre a megfelelő érték térül vissza.

1. Copy konstruktor

Ez a teszteset egy 3×3-as mátrixot olvas be az *input2.txt* fájlból, majd a copy konstruktort meghívja egy második mátrixra. Ezt követően ellenőrízzük, hogy mindkét mátrix ugyanazon indexeire ugyanaz az érték térül vissza.

1. Értékadás operátor

Ez esetben létrehozunk két 3×3-as mátrixot: A és B. Az A-t beolvassuk az *input2.txt* fájlból, majd elvégzünk egy értékadást (A=B). Ezt követően ellenőrízzük, hogy mindkét mátrix ugyanazon indexeire ugyanaz az érték térül vissza.

1. Összeadás

Ebben a tesztesetben az *input3.txt* fájlból három 4×4-es mátrixot olvasunk be, majd ezeket összeadjuk és ellenőrízzük azt, hogy a várt eredményt kapjuk, majd az összeadás két tulajdonságát: a kommutativitást és asszociativitást.

1. Szorzás

Ebben a tesztesetben szintén az *input3.txt* fájlból olvasunk be három 4×4-es mátrixot, majd ellenőrízzük, hogy a várt eredményt kaptuk-e, majd ellenőrízzük a mátrxiszorzás asszociatív tulajdonságát.

1. Hibakezelés

Az utolsó tesztesetben olyan műveleteket végzünk, amelyek hibát kell kidobjanak és a *try-catch* blokkokkal ellenőrízzük, hogy a program megtalálja-e a hibát és a megfelelő hibanevet adja vissza.