

Listakezelés haladó szinten

A LOGO nyelv nagy hiányosságának tartják sokan, hogy nem rendelkezik tömb adattípussal. Ez a szó klasszikus értelmében igaz is, de ha speciális felépítésű listákat készítünk, akkor a vektor, mátrix és tetszőleges dimenziójú tömb is előállítható és kezelhető az Imagine-ben.

Vektor létrehozása és kezelése

Egy lista tetszőleges elemét tudjuk megjeleníteni a **mélyelem** paranccsal. Ennek használatakor jó tudni, hogy vektorként kezeli a lista elemeit 1-től indexelve. Ha a vektor határain túllépünk, akkor az üres listát adja vissza.

```
? mutat mélyelem 3 [a b c d]
c
? mutat mélyelem 6 [a b c d]
[]
? mutat mélyelem 0 [a b c d]
[]
```

A vektorba írni is tudunk. Erre szolgál a **mélycserél** parancs.

```
? mutat mélycserél 4 [a b c d e f] "d2
[a b c d2 e f]
```

Ha az indexhatárokon kívülre mutat az első hely paraméter, akkor a vektor nem változik meg.

```
? mutat mélycserél 8 [a b c d e f] "d2
[a b c d e f]
```

Vegyük észre a különbséget a **csere** és a **mélycserél** parancs között!

```
? mutat csere 8 [a b c d e f] "d2
[a b c d e f d2]
```

Kétdimenziós tömb létrehozása és kezelése

Az azonos hosszúságú listákból felépülő listákkal kétdimenziós tömböt tudunk megadni. A jobb oldali 4×3-as mátrix például megadható

```
[[1 2 3 4][5 6 7 8][9 10 11 12]]
```

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$$

lista formájában.

A **mélyelem** és **mélycserél** parancsok első paramétere lista is lehet. Ebben az esetben az első lista első eleme a főlista elemét azonosítja, az első lista második eleme már az allistán belüli elemet.

```
? mutat mélyelem [2 3] [[1 2 3 4][5 6 7 8][9 10 11 12]]
7
```

Hasonlóan használható a **mélycserél** eljárás is. Ha például a mátrix 2. sorában lévő 3. elemet akarjuk cserélni, akkor a

```
? mutat mélycserél [2 3] [[1 2 3 4][5 6 7 8][9 10 11 12]] "d
[[1 2 3 4] [5 6 d 8] [9 10 11 12]]
```

Itt is fontos megjegyeznünk, hogy a változást tárolni is kell, a második bemenetként megadott lista ténylegesen nem módosul.

*22. Készítsünk egy **tömb.létrehoz** nevű függvényt, amely három paraméterrel rendelkezik: az első a tömböt alkotó sorok, a második az oszlopok száma. A tömb elemeit alkotó elem legyen a harmadik paraméter!*

A függvény megvalósításához használjuk fel a **lokálisérték** parancsot. Elsőként a sorokat, majd az oszlopokat töltjük fel.

```
eljárás tömb.létrehoz :i :j :e
  lokálisérték "l []
  ism :j [lokálisérték "l elsőnek :e :l]
  lokálisérték "m []
  ism :i [lokálisérték "m elsőnek :l :m]
  eredmény :m
vége
```

A függvény használatát mutatja a következő példánk:

```
? mutat tömb.létrehoz 2 5 "#
[[# # # # #] [# # # # #]]
```



Ne felejtsük el, hogy a létrehozott tömb alapvetően kétdimenziós, így az alábbi eredményeket kapjuk, ha speciális sor- és oszlopszámot adunk meg.

```
? mutat tömb.létrehoz 1 5 "#
[[# # # # #]]
? mutat tömb.létrehoz 1 1 "#
[[#]]
? mutat tömb.létrehoz 1 0 "#
[[[]]]
? mutat tömb.létrehoz 0 0 "#
[]
? mutat tömb.létrehoz 0 5 "#
[]
```

A feltöltő elem persze lista is lehet.

58. Alakítsuk át az eljárást úgy, hogy ha a sor vagy oszlop paraméterek közül az egyik 1, akkor csak egyszerű listát hozzon létre!

Vegyük észre, hogy ha lenne egy **vektor.létrehoz** nevű, elemszámot és elemet tartalmazó függvényünk, akkor azzal a **tömb.létrehoz** helyettesíthető lenne!

*59. Készítsük el a **vektor.létrehoz** nevű, elemszám és egy elem paraméterű függvényt!*