

2. GYAKORLAT

NEMNUMERIKUS TÍPUSOK, MÁTRIXOK

KARAKTERLÁNCOK

A Matlab az írásjeleket, karaktereket 2 bájtól tárolható numerikus értékekkel kódolja. Ha több írásjelet fűzünk össze, akkor karakterláncról beszélünk.

```
>> x = 'alma'  
>> y = x+0, whos x y, x'
```

Egy karakterlánc valójában numerikus értékeként is kezelhető elemeket tartalmazó sorvektort jelent. Ez abból is látszik, hogy egy karakterláncsal, mint sorvektorral műveleteket is végezhetünk. Ha egy karakterkódokat tartalmazó vektorra a *char* függvényt alkalmazzuk, akkor karakterláncot kapunk.

(Ha a *char* függvényt eleve karakterláncra alkalmazzuk, akkor nem történik változás.)

Feladat

Adjuk meg azt a karakterláncot, amelynek karaktereit a következő ASCII kódsorozat azonosítja:
w = [104 101 108 108 111 32 119 111 114 108 100 33]

Több karakterlánc egy sorvektorba elhelyezve, azok összeragasztását eredményezi:

```
>> p = 'Maci'; q = 'Laci'; [p q]
```

Feladat

Az előző w sztring végéről hagyjuk el az utolsó kódot, majd a '!' karakterrel történő összeragasztással készítsük el újra a teljes eredeti karakterláncot.

Megj.: `char(w1, '!')` nem ad jó megoldást, `char([w1, !])` pedig szintaktikailag hibás.

A *char* függvény paraméterében megadott skalárokkal – mint számértékekkel – műveletek is végezhetők.

Feladatok

Állítsuk elő az `int8` adattípus legnagyobb értékénél eggyel kisebb ASCII kódú karaktert!

Írassunk ki különböző magyar ékezetes karaktereket ASCII kódok alapján!

(Segítség: utóbbihoz érdemes egy számvektor elemeit egyszerre karakterizálni, pl. 190:230.)

A karakterláncokhoz kapcsolódó további példákat és feladatokat javasolt házi feladatként megnézni (lásd: Otthoni munka)!

DÁTUM-IDŐ ADATOK

Feladat

Adjunk választ a Matlabbal – esetleg a súgót is felhasználva – a következő kérdésekre:

Mi az 1000. január 1. dátum sorszáma a Matlab rendszerben?

Hány naposak vagyunk ma?

(Tipp: a *now* ugyanúgy működik, mint az Excelben a *Most* függvény. A törtrész elhagyását a *floor* függvénnyel tudjuk elérni.)

Gyakran van szükségünk a számítások közben eltelt idő mérésére. Ehhez a fontosabb eszközök:

now – aktuális dátum-idő (törtrész a napon belüli idővel arányos)

cputime – a hasznos számításokkal eltöltött idő

tic, toc – a CPU órajeleinek száma.

A tényleges számítási idő a *cputime* segítségével mérhető, a ciklusszervezési időt levonjuk!

```
>> x0 = sqrt(2); z = 1e8; dx = 1.0000001;
>> t = cputime(); for i = 1:z end, alap_ido = cputime()- t
>> x = x0; t = cputime(); for i = 1:z x = x+dx; end, plus_ido =
cputime()- t
>> x = x0; t = cputime(); for i = 1:z x = x/dx; end, div_ido =
cputime()- t
>> hanyados = (div_ido - alap_ido)/(plus_ido - alap_ido)
```

Feladat

Mérjük le a fentieket a *tic, toc* páros segítségével is.

A *tic* a CPU ütemszámát kérdezi le az indítás(telepítés) óta. Az ütemszámot egy 64 bites tárolóban szaporítja. Ezzel csak szimbolikusan lehet számolni, viszont a *cputime* paranccsal kapott eredmény *double* típusú lesz.

KIFEJEZÉSEK, HASONLÍTÁSOK

Feladat

Próbáljunk ki egyszerű példákat a szöveges (és numerikus) adatok hasonlításáról és a velük való műveletvégzésről. Például:

```
>> 'al' + 'ma' % összeadás karakterenként, numerikus eredmény
>> 'al' < 'ma' % hasonlítás karakterenként, logikai eredmény
>> 'alma' + 'fa' % az összeadás nem végezhető el (méréthiba)
>> '12' + '23' % összeadás karakterenként, '1' + '2' = 'c' !!!
>> char(49+50) % magyarázat '1' + '2'-höz
>> '1' + '2' == '3' % nyilván hamis
```

Önállóan gyakoroljunk tovább. A tényleges kipróbálás előtt írjuk le a füzetünkbe a várható eredményt! (Mindig gondoljuk végig, hogy a hasonlítás, ill. a megfelelő művelet elvégezhető-e, és milyen típusú lesz az eredmény.)

Pl. 'Kovács' < 'Kovácsné'; 'Kovács ' < 'Kovácsné';

'12' + '12'; '12' < '23'; '12' == '23'; '12' + 12; '12' + '23' == '35'

Vigyázzunk arra, hogy egyes esetekben teljesen más a művelet értelmezése, mint amit esetleg a tapasztalatból várhatunk (ismert programozási környezetek)!

Pl. 'al' & 'ma'!

Módosítsuk a feladatot úgy, hogy bevetünk egyszerűbb függvényeket is! (Pl.: *sum*, *max*, *size*.)

VEKTOROK, MÁTRIXOK LÉTREHOZÁSA, HIVATKOZÁSOK

A Matlab az egymás mellett felsorolt adatokat sorvektornak (egysoros mátrixnak), az egymás alatt felsorolt adatokat oszlopvektornak (egyoszlopos mátrixnak) tekinti. Amikor egy vektort elemeivel megadunk, szögletes zárójelbe kell azokat foglalnunk (ez azonban időnként elhagyható, próbáljuk ki):

```
>> w = [2 sin(pi/6) 'a'-'A'], z = [0:3]'  
>> s = [1:3:20], q = linspace(0,30,16)
```

Feladatok

Állítsuk elő a kettőspont operátor felhasználásával a

j = 1 4 7 10 13 16 19

sorvektort!

(Ha több megoldás is létezik, akkor azt adjuk meg, amelyiknél a felső határ a legkisebb.)

A : operátor felhasználásával töltsük fel a „k” sorvektort a kétjegyű pozitív páros számokkal!

Állítsuk elő a *linspace* parancs felhasználásával az

s = 0 30 60 90 120 150 180 210 240 270 300

sorvektort!

Állítsuk elő a *linspace* parancs felhasználásával a kétjegyű pozitív számokat tartalmazó „a” sorvektort!

A mátrix (téglalapba rendezett adatok) megadása szintén az elemeinek felsorolásával is történhet, a soron belüli delimiter (elválasztó) vagy a szóköz, vagy a vessző írásjel. Egy sor végét a pontosvessző jelzi (az utolsó után nincs pontosvessző!). Vigyáznunk kell arra, hogy minden sor ugyanannyi adatot tartalmazzon!

(Különböző típusú adatok együtt történő alkalmazásáról lásd az Otthoni munka részt!)

A vektorokat hagyományosan kisbetűvel kezdődő változóval, a mátrixot nagybetűvel kezdődő változóval azonosítjuk. (Így áttekinthetőbb írásmódhoz jutunk.)

Feladatok

Egy „c” változóba rakjuk be az első 4 pozitív páros számot (növekvően) úgy, hogy sorvektor jöjjön létre!

Egy „d” változóba rakjuk be az első 4 pozitív páratlan számot (növekvően) úgy, hogy oszlopvektor jöjjön létre!

Töltsük fel *aritmetikai műveletek és függvények segítségével* csupa 5 elemmel az A 3×2-es méretű mátrixot!

(Felesleges zárójeleket és szimbólumokat ne használjunk a fenti feladatokban.)

A 3×4 méretű X mátrixot töltsük fel soronként a fok = [0, 15, 30, 45] szögértékekkel, ezek tangenseinek és kotangenseinek értékeivel! (fok–radián átváltás!)

A mátrixok és vektorok létrehozásához használhatjuk a Variable Editort, ill. lehetőség van az adatok egyes külső fájlokból (pl. Excel munkafüzet) történő beimportálására is.

Feladatok

A Variable Editor segítségével állítsuk elő azt a B azonosítójú 5×5-ös méretű mátrixot, amelynek jobb alsó sarkában az [1 2; 3 4] mátrix van, a többi eleme pedig 0.

Hajtsuk végre ezt a feladatot a *clear B* parancs kiadása után értékadással is!

Olvassuk be az *adat.xls* munkafüzet *valami* lapjáról az F5:H8 tartományt a C mátrixba!

Az elemek felsorolásakor már létező mátrixra/vektorra, műveletekkel kiszámított értékekre is hivatkozhatunk, sőt a szögletes zárójelekkel történő megadások egymásba is ágyazhatók.

```
>> A = [1:4; 2 4 6 8]
>> x = [0:30:180]; W = [x; sin(pi*x/180); cos(pi*x/180)]
% radiánba váltjuk (alternatíva: sind, cosd)
>> S = [x' sin(pi*x/180)' cos(pi*x/180)']
>> T = [[x', sin(pi*x'/180)] cos(pi*x/180)']
% az almátrixoknak illeszkedniük kell
```

Mátrixok elemeire a következő módon hivatkozhatunk:

A(s_ind, o_ind) ill. A(sorszám) % a megadott pozíciójú elem, ill. oszlopfolytonosan a megadott sorszámú elem

A(s_ind, :) % teljes sor

A(:, o_ind) % teljes oszlop

A(s_ind_1:s_ind_2, :) % néhány sor (s_ind_1 és s_ind_2 határokkal)

A(s_ind_1:s_ind_2, o_ind_1:o_ind_2) % részmátrix a megfelelő határokkal

Feladatok

Legyen A = [1:4; 5:8; 9:12; 13:16]. Adjuk meg a megfelelő hivatkozással

- a 7 elemet (pozícióval és oszlopfolytonosan is);
- A második sorát;
- A harmadik oszlopát;
- A első oszlopát sorvektorként;
- a [6 7; 10 11] részmátrixot;
- a [11; 15] részmátrixot.

A részmátrix-hivatkozások is felhasználhatók mátrixok összeillesztésére, azaz a fentiekhez hasonló típusú feladatok megoldására.

Feladatok

Legyen $A = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$ és $B = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}$.

Állítsuk elő az A és a B felhasználásával a

C =

1	2
1	2
3	3
3	3
3	3

mátrixot.

Állítsuk elő az A és a B felhasználásával a

C =

1	1
2	2
3	3

mátrixot.

(Felesleges zárójeleket és szimbólumokat ne használjunk a fenti feladatokban.)

Feladat

Az A, B, C, D, E mátrixokról a következőket tudjuk:

- $\text{size}(A) : [4 \ 5]$
- $\text{size}(D) : [3 \ 5]$
- $E = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ és négyzetes

Milyen méretűek a B és C mátrixok?

Szorgalmi feladat

Ellenőrizzük az $A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 3 \\ 3 & 3 & 3 \end{bmatrix}$ mátrix determinánsát az első sor szerinti kifejtési tétel alkalmazásával!

Megoldás

```
>> A = [1:3;2:4;3 3 3], detA = det(A)
>> A1 = A(2:3,2:3)
>> A2 = A; A2(1,:) = []; A2(:,2) = []
% A2 = A([2:3],[1,3]) módon is megadható
>> A3 = A(2:3,1:2)
% A1, A2, A3 első sor szerinti minormátrixok
>> dA = A(1,1)*det(A1)-A(1,2)*det(A2)+A(1,3)*det(A3)
% kifejtési tétel
```

PONTOZOTT MŰVELETEK

A Matlab az azonos méretű mátrixok(vektorok) között megengedi az elempáronkénti műveleteket. Ez hasonló az Excel blokkok(tartományok) között elempáronként elvégezhető műveletekhez. Ezek a következők:

- összeadás, kivonás (nem szabad '.' jelet a + illetve a – elé írni!)
- szorzás, osztás, hatványozás (itt kötelező az elempáronkénti műveletek előtt a '.' jelet használni!)

Ha az egyik operandus skalár, akkor ez valójában a másik operandusnak megfelelő méretű és csupa azonos elemből álló mátrixot jelent.

A pontozott transzponálás és a sima transzponálás között csak annyi a különbség, hogy a komplex elemek konjugálása az elsőnél nem történik meg, sima transzponáláskor viszont igen.

Példák, feladat

a./ Legyen $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, ekkor

$B = A - 3$ valójában az $A - \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix}$ műveletet jelenti.

b./ $B = A.^{-1}$ valójában az $A.^{[-1 \ -1; -1 \ -1]}$ műveletet jelenti, azaz az B mátrix elemei az A mátrix elemeinek reciprokjai lesznek.

c./ Ha $B = \begin{bmatrix} 3 & 2 \\ 4 & 1 \end{bmatrix}$, akkor $C = A.*B$ mátrix $\begin{bmatrix} 1*3, 2*2; 3*4, 4*1 \end{bmatrix}$ azaz $\begin{bmatrix} 3 & 4 \\ 12 & 4 \end{bmatrix}$ lesz.

Kérdés: Kommutatív a pontozott szorzás?

d./ $A = \begin{bmatrix} 8 & 1 & -5; 3 & 13 & 7; 1 & 2 & 34.5 \end{bmatrix}$

Szimmetrikus-e a $B = A + A'$ mátrix?

Mi a válaszuk akkor, ha $A(1, 3) = -5 - i$, azaz az A mátrix a főátlón kívül tartalmaz komplex elemet?

Feladat

Adott a C és az $A = \begin{bmatrix} 1 & 5; 2 & 3 \end{bmatrix}$ mátrix. Állítsuk elő a B mátrixot, ha ...

a./ $C = \begin{bmatrix} 2 & -5; 2 & -6 \end{bmatrix}$ és $C = A.*B$

b./ $C = \begin{bmatrix} 0.5 & -5; 2 & -1.5 \end{bmatrix}$ és $C = A./B$

Ellenőrizzük az eredményeket!

A megoldások: $(C./A)$ és $(A./C)$

A pontozott műveleteket elsősorban a függvények kiszámításánál használjuk. Ilyenkor a pont nélküli kifejezés hibaüzenetet generálhat (ill. nem feltétlenül az történik, amit szeretnénk)!

Feladat

Legyen $x = 0:\pi/9:\pi$ (sorvektor). Hajtsuk végre a Matlabban a $\sin(x)/x$ és a $\sin(x)./x$ műveleteket és magyarázzuk meg az eredményt!

(Segítség: a második eredmény nyilvánvaló, az elsőnél azt kell meggondolni, hogy x az 10 elemű sorvektor, a pinv(x) pedig 10 elemű oszlopvektor.)

OTTHONI MUNKA

Feladat (szöveges adatok)

Állítsuk elő minél több módon az 'ABC' karakterláncot!

Megoldás

```
>> 'ABC', ['A' 'B' 'C'], ['A','B','C'], char(65,66,67)',  
char('abc' - 32) % és így tovább ...
```

Feladat

Állítsuk elő a kettőspont sorozatképző operátorral (lásd még lent is) az angol ábécé nagybetűinek sorvektorát, majd a karakterláncot is. Ebből kiindulva állítsuk elő

a./ a kisbetűk karakterláncát!

b./ a nagybetűs karakterlánc palindromját (tükörképét)!

(Tipp: itt használjuk ki, hogy az angol ábécé egy karakterének és tükörkép karakterének a kódösszege mindig ugyanaz az érték; 155.)

Megoldás

a./

```
>> U = char('A':'Z'), diff = 'a' - 'A', L = char(U+diff)  
>> whos U L
```

Megjegyzés: az $L = \text{char}((65:90) + 32)$ is jó lett volna.

Látható, hogy a kisbetűk kódja 32-vel nagyobb, mint a nagybetűké.

(Próbáljuk ki a *lower/upper* parancsokat is.)

b./

```
>> palindrom_U = char('A' + 'Z' - U) % vagy char(155 - U)
```

Feladat (szöveges adatok)

Állítsuk elő az 'áÁéÉíÍóÓöÖóóúÚűÜűű\$€' sztring karaktereinek az ASCII kódját!

Rendezzük a karakterláncot kódok szerint növekedően! (Eml.: *sort* parancs.)

Feladat (szöveges adatok)

Állítsuk elő az a = 'boci '; b = 'tarka'; d = 'se '; e = 'füle ' karakterláncokból az ismert kétsoros versikét! Mi az első sor utolsó elemének kódja?

Megoldás

```
>> a = 'boci '; b = 'tarka'; c = b; c(1) = 'f';  
>> d = 'se '; e = 'füle '; % a szavak  
>> vers = char([a a b], [d e d c])  
% a két sor szavakból összerakva  
>> vers(1,length(vers))+0 % első sor utolsó jelének kódja
```

Megjegyzés: Ha a *char* függvényt több és vesszővel elválasztott karakterláncra alkalmazzuk, akkor sorokra tördelt egyesített karakterláncot kapunk. Itt a rövidebb sorok annyi szóközzel egészülnek ki, hogy a hosszuk egyenlő legyen a leghosszabb sor hosszával (nálunk most a 15 betűs első sor kiegészült 16 írásjelre).

Feladat (adattípusok)

Nézzük meg, hogy hogyan működik az *isnumeric*, *islogical*, *isscalar* és az *isvector* függvény! (Súgó.)

Feladat (komplex számok)

Az előadás fóliák és a súgó segítségével idézzük fel, hogy a Matlab hogyan tudja megvalósítani a komplex számokkal történő műveletvégzést. Keressünk példafeladatokat (pl. matek jegyzetek/tankönyvek, zh feladatok), és ellenőrizzük a megoldásokat!

(Például: legyen $z_1 = 3 + 5i$, $z_2 = 4 - 2i$. Ellenőrizzük, hogy $z_1/z_2 = 0,1 + 1,3i$.)

Nézzük meg, hogy hogyan működik az *isreal* függvény!

Feladat (időmérés)

Olvassuk el a Súgóban a „Programozási alapok” című részben a programok futásidejének méréséről és elemzéséről szóló részt (tic-toc vs. cputime). Írjuk le a tapasztalatokat a füzetünkbe. Próbáljuk ki időmérésre a *clock* és az *etime* parancsokat is.

Feladat (mátrixépítés)

A fenti $A = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$ és $B = \begin{bmatrix} 3 & 3 \\ 3 & 3 \\ 3 & 3 \end{bmatrix}$ mátrixok, ill. megfelelő darabjaik felhasználásával készítsünk további C mátrixokat!

Feladat (mátrixépítés, különböző típusok!)

Ha a mátrixok/vektorok megadásánál az elemek típusa különböző, akkor ez az elemek konverziójához vezet, ami túlcsondulást is eredményezhet:

```
>> A = [eps*1e16, pi; 2, floor(now())/2014] % mind double
>> B = [eps*1e16 int8(pi); 2 365.286] % mind int8, túlcsondulás
>> C = [90 72 45 237 114 450/2 's'] % mind char típusú lett
>> P = [tic pi; eps*1e16 i] % mind komplex uint64 lett
(tic miatt)
```

Ellenőrizzük a fenti mátrixok és vektorok típusát!

Látható, hogy egy mátrix minden eleme azonos tárolási osztályba tartozik, akárhogyan is adtuk meg eredetileg! Tanulság: a mátrix elemeinek megadásakor lehetőleg ne keverjük az elemek típusát, mert ugyanis konverzió következik be, és ez többnyire információvesztéssel járhat.

Feladat

Az A, B, C, D, E mátrixokról a következőket tudjuk:

- `size(A) : [4 5]`

- `size(D)` : [3 5]
- `E = [A B; C D]` és négyzetes

Milyen méretűek a B és C mátrixok? (Próbáljunk először Matlab nélkül választ adni a kérdésre.)

Feladat (átméretezés)

A *reshape* függvényt átméretezésre használhatjuk (többféle módon is). Az elemek száma nem változik, a mátrix méretei – természetesen – igen.

1. Legyen `C = reshape(1:15, 3, 5)`. Adjuk meg a 13 elemet jelentő hivatkozást!
(Segítség: a *reshape* függvény a sor/oszlop-vektort oszlopfolytonosan tördeli)
2. Töltsük fel sorfolytonosan a 0...99 számokkal a D 10×10 méretű mátrixot!
(Megoldás: `D = reshape(0:99,10,10)'`)

Feladat (pontozott műveletek)

Készítsünk kifejezéseket pontozott műveletek felhasználásával oly módon, hogy minél többféle módon bemutadjuk a következő lehetőségeket:

- A helyes működéshez ténylegesen szükséges a pont jel.
- A pont jel kirakása felesleges, de nem okoz hibát.
- A pont jel kirakása hibát okoz!

