

2. GYAKORLAT

NEMNUMERIKUS TÍPUSOK, MÁTRIXOK

KARAKTERLÁNCOK

A Matlab az írásjeleket, karaktereket 2-bájton tárolható numerikus értékekkel kódolja. Ha több írásjelet fűzünk össze, akkor karakterláncról beszélünk. Egy billentyűzetről közvetlenül bevihető karakterláncot a következőképp adhatunk meg:

```
>> x = 'áÁéÉíÍóÓöÖőŐúÚüÜűŰş€'  
>> y = x+0, whos x y, x'
```

(itt az euro jelét AltGr+U-val vittük be). Egy karakterlánc valójában numerikus értékeként is kezelhető elemeket tartalmazó sorvektort jelent. Ez abból is látszik, hogy egy karakterláncsal, mint egy 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]
```

A char típusú skalárokkal mint számértékekkel művelet is végezhető.

Feladat

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

Feladat

Állítsuk elő az sorozatképző operátorral 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.

b./

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

Ha a `char()` függvényt több és vesszővel elválasztott karakterláncra alkalmazzuk, akkor sorokra tördelt egyesített karakterláncot kapunk. Itt a rövidebb sorok annyi space-el egészülnek ki, hogy a hosszuk egyenlő legyen a leghosszabb sor hosszával.

Feladat

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 az $a = \text{'boci '}$; $b = \text{'tarka'}$; $d = \text{'se '}$; $e = \text{'füle '}$ 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: a 15 betűs első sor kiegészült 16 írásjelre, mert a második sor 16 jelet tartalmaz.

Feladat

Próbáljunk ki a VBA résznél bemutatottakhoz hasonló egyszerű példákat a szöveges adatok hasonlításáról és a velük való műveletvégzésről. 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. 'al' + 'ma' ; 'al' < 'ma' ; 'alma' + 'fa' ; $\text{'Kovács' < 'Kovácsné'}$; $\text{'Kovács' < 'Kovácsné'}$;
 '12' + '12' ; '12' < '23' ; '12' + 12 ; $\text{'12' + '23' == '35'}$

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 törtész elhagyását a `floor()` függvénnyel tudjuk elérni.)

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

`now()` – aktuális dátum-idő (törtész 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` lesz.

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:

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

Feladatok

Állítsuk elő a : 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.)

Állítsuk elő a linspace parancs felhasználásával az

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

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

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 mátrixváltozóba 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!

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

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

Feladat

Ellenőrizzük a fenti mátrixok és vektorok típusát!

Jegyezzük meg: egy mátrix minden eleme azonos tárolási osztályba tartozik, akár hogyan is adtuk meg! A mátrix elemeinek megadásakor lehetőleg ne keverjük az elemek típusát, mert úgyis konverzió következik be, és ez többnyire információvesztéssel járhat.

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

```
>> A = [1:4; 2 4 6 8]
>> x = [0:30:180]; W = [x; sin(pi*x/180); cos(pi*x/180)]
>> 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 $[6\ 7; 10\ 11]$ részmátrixot;
- a $[11; 15]$ részmátrixot.

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

Feladatok

Legyen $A = [1\ 1; 2\ 2]$ és $B = [3\ 3; 3\ 3; 3\ 3]$.

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:

- `size(A)` : [4 5]
- `size(D)` : [3 5]
- `E = [A B; C D]` és négyzetes

Milyen méretűek a B és C mátrixok?

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

Feladatok

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)'`)

Ráadás feladat

Ellenőrizzük az `A = [1:3; 2:4; 3 3 3]` 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
```

OTTHONI MUNKA

Az előadás fóliák és a sűgó 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 $z_1/z_2 = 0,1 + 1,3i$.)

Olvassuk el a Sógóban 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.

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!

