Matlab programozás elemei

stringek:

’ ’ között megadott karaktersorozat. Vektorként kezelhető:

```
>> s='ez egy string'

s =

ez egy string

>> length(s)

ans =

    13

>> s(2)

ans =

z

>> s(2:5)

ans =

z eg

>> [s(1:7) 'hosszabb ' s(8:13)]

ans =

ez egy hosszabb string
```

kiiratás:

Ha egy parancsot nem zárunk le pontosvesszővel, akkor az outputja automatikusan kiíródik a képernyőre.

Vektor/mátrixok formázott kiiratását a **`disp`** paranccsal hajthatjuk végre a legegyszerűbben:

```
>> A=[2 4 5; -1 2 3]

A =

     2     4     5
    -1     2     3

>> disp(A)
     2     4     5
    -1     2     3
```

A `disp` paranccsal lehet pl. stringeket is kiiratni:

```
>> disp('lklkjlkj')
lklkjlkj
```

Másik formázott kiiratási lehetőség az fprintf parancs használata. Szintaxisa a megfelelő C parancséval gyakorlatilag azonos:

```
>> fprintf('Ez az %d. példa\n',1)
Ez az 1. példa
```

A kiiratás alapértelmezésben néhány tizedesjegy pontosságú. Ezt a következő paranccsal tudjuk módosítani:

```
>> format long
>> sqrt(2)

ans =

   1.41421356237310
```

Ciklusszervező parancsok

Használhatók parancsmódban is.

```
for i=1:4,
  i
end
```

```
i =
     1
i =
     2
i =
     3
i =
     4
```

```
ind=5:-2:-8
for k=ind,
   k
end
```

```
ind =
     5     3     1    -1    -3    -5    -7
k =
     5
k =
     3
k =
     1
k =
    -1
k =
    -3
k =
    -5
k =
    -7
```

```
k=1;    % nem írja ki az outputot, ha ';'-vel zárjuk le a sort
while k<10,
   k=k+3
end
```

```
k =
     4
k =
     7
k =
    10
```

logikai operátorok:

&	és
|	vagy
~	nem

relációs operátorok:

< > <= >= == ~=

Parancsfájlok (script):

Egy .m kiterjesztésű fájlban tetszőleges Matlab parancsokat sorolhatunk fel. Ezeket egymás után végrehajtja meghíváskor. A parancs neve a fájl neve (kiterjesztés nélkül).

Függvények, programok szintaxisa

Ha egy .m kiterjesztésű fájl a

```
function [out1,…,outm]=fgvnev(input1,…,inputn)
```

alakú sorral kezdődik, akkor a fájl egy Matlab függvényt definiál. Ekkor a fájl neve és az első sorban megadott *fgvnev*-vel legyen azonos! (Ha nem az, a fájl neve lesz a függvény neve.)
Függvényeknek/eljárásoknak lehet több input ill. output változója, de bármelyik elhagyható.
Ha egy output változója van a függvénynek, nem kell a [ ] az output változó körül.
Az output változóknak értéket kell adni az eljáráson belül.

```
function y=fg1(x)
if x < 2,
    y=x^4-1;
else
    y=5*x+2;
end
```

```
>> fg1(0)

ans =

    -1

>> fg1(3)

ans =

    17

>>
```

Az `if` parancs

```
if feltétel,
    utasítás1
elseif feltétel2,
    utasítás2
...
else
    utasításn
end
```

Itt a feltételx lehet logikai kifejezés, illetve egész értékű kifejezés is. (0 : hamis, nem 0 : igaz)

Egy sorban is írhatjuk a parancsot, de ekkor vesszővel vagy pontosvesszővel kell elválasztani az utasításokat:

```
>> if 0, fprintf('igen'), else fprintf('nem'), end

nem

>> if 1, fprintf('igen'); else fprintf('nem'); end

igen

>> if 2, fprintf('igen'), else fprintf('nem'), end

igen
```

`break` parancs:

`for` vagy `while` ciklusból kiugrik

`continue` parancs:

`for` vagy `while` ciklusban a feltétel kiértékelésére ugrik vissza

`return` parancs:

Az adott eljárás/függvény futása véget ér.
Nem kell használni az eljárás végén.

`switch` parancs:

```
switch kifejezés
      case érték1
            utasítás1
      case érték2
            utasítás2
     ...
     otherwise
           utasításn
end
```

Kiértékeli a `kifejezés`-t, amely skaláris vagy string lehet, és ha az értéke megegyezik `érték1`-gyel, akkor az `utasítás1` hajtódik végre, ekkor az utána leírt többi eset nem hatódik végre, stb.
Több értéket megadhatunk egy ágban, ekkor { } között soroljuk fel az értékeket.

```
% példa switch használatára, parancsfájl neve: pl1

x=[-1,3,5,0,-2]
for v=x,
    fprintf('x=%d:  ',v)
    switch v
    case {-1,-2,-3,-4}
        fprintf('negatív\n')
    case 0
        fprintf('nulla\n')
    case {1,2,3,4}
        fprintf('pozitív\n')
    otherwise
        fprintf('nem szerepel\n')
    end
end
%----------------------------------------
```

```
>> pl1

x =

    -1     3     5     0    -2

x=-1:  negatív
x=3:  pozitív
x=5:  nem szerepel
x=0:  nulla
x=-2:  negatív
```

Cella adattípus:

A cella (cell) vektor/mátrix-hoz hasonló adattípus, csak eltérő típusú és hoszú adatok tárolására használható. Kerek zárójel helyett { } között adjuk meg az indexeket:

```
>> a{1,1}=1

a =

    [1]

>> a{1,2}='szoveg'

a =

    [1]    'szoveg'

>> a{2,1}='asdasdasd'

a =

    [        1]    'szoveg'
    'asdasdasd'          []

>> a{2,2}=1+j

a =

    [        1]    'szoveg'
    'asdasdasd'    [1.0000+ 1.0000i]

>> c={'asdf','sdfsdf sdfs k','oo ll pp'}

c =

    'asdf'    'sdfsdf sdfs k'    'oo ll pp'

>> c{2}

ans =

sdfsdf sdfs k
```

Struktúrák:

A következő szintaxissal kezelhetünk struktúrákat:

```
>> A.nev='Nagy Balázs'

A =

    nev: 'Nagy Balázs'

>> A.szev=1990

A =

     nev: 'Nagy Balázs'
    szev: 1990

>> A.szho='január'

A =

     nev: 'Nagy Balázs'
    szev: 1990
    szho: 'január'

>> A.sznap=1

A =

      nev: 'Nagy Balázs'
     szev: 1990
     szho: 'január'
    sznap: 1
```