MATLAB/OCTAVE ALAPOZÓ FOLYTATÁS

FORMÁZOTT SZÖVEGEK KIÍRÁSA

Gyakran van szükség arra, hogy az eredményeinket egy adott formátumban jelenítsük meg. Nézzük például a szögekkel való műveleteket. A legtöbb matematikai művelet végzésére alkalmas szoftver (pl. Matlab, Octave, Excel…) a szögeknél a radiánt tekinti alapértelmezettnek, ha ettől eltérő formátumban szeretnénk látni az eredményeket, akkor nekünk kell erről gondoskodni. Matlab és Octave alatt a radiánban értelmezett trigonometriai függvényeknek (pl. sin, cos, tan, atan, atan2…) megvannak a fokokkal számoló változatai (pl. sind, cosd, tand, atand, atan2d…), de ha már fok-perc-másodpercben szeretnénk megjeleníteni az eredményeket (pl 302-06-23), esetleg adott számú tizedesjegyre (23-03-48.5831), akkor ezt a formázott szövegekkel tehetjük meg. Ugyanígy, ha pl. képeket szeretnénk automatikusan elnevezni egy ciklusban pl. IMG0001, IMG0002 stb. akkor ehhez is a formázott szövegeket hívhatjuk segítségül.

Az `fprintf` paranccsal a fájlba írhatunk formázott szövegeket vagy a képernyőre, az `sprintf` használatával pedig egy stringbe (szöveges változóba)/képernyőre. A következő formátumjelölőket alkalmazhatjuk:

- %d – egész szám, %s – szöveg, %f – valós szám (lebegőpontos), %c – karakter, %u – nem előjeles egész
- %e – normál alak pl. 3.14e+00, %E – 3.14E+00
- %g – kompakt forma, %f vagy %e közül a rövidebbik, fölös 0-k nélkül

A típust jelző betű előtt szerepelhet még pl. + jel, akkor előjelesen írja ki a számot; mező szélesség; tizedesjegyek száma; 0, akkor 0-kal tölti fel elől az üres helyeket a mező szélességéig.

Próbáljuk ki a következőket!

```
> fprintf('pi')
> sprintf('pi')
> sprintf('%f',pi)
> sprintf('%.2f',pi)
> sprintf('%6.2f',pi)
> sprintf('%+6.2f',pi)
> sprintf('%0+6.2f',pi) → +03.14
```

A legutolsó kifejezés egy valós szám lesz (f), ahol a mező szélessége 6 (tizedespontot is beleértve!), a tizedesjegyek száma 2 (6.2), kiírja az előjelet (+) és 0-kat a mező szélességéig az üres helyekre (0), hogy meglegyen a 6 széles mező (0 nélkül szóközökkel töltené fel), így írja ki a pi értékét.

Ha az eredmény hosszabb, mint a mező szélessége, akkor nem veszi figyelembe a megadott mező szélességet. Pl.

```
> sprintf('%2.5f',1/eps) → 4503599627370496.00000
> sprintf('%2.5g',1/eps) → 4.5036e+015
```

Írjunk egy saját programot, ami a tizedfokban megadott eredményünket a geodéziában szokásos fok-perc-másodperc formában írja ki. A perc, másodperc értékét mindig két számjeggyel írja ki pl. 192-03-12.

```
> function str = fpm(x);
> % A fuggveny tizedfokbol fok-perc-masodperc ertekekbe szamol at.
> % A kimenet egy formázott szöveg (ddd-mm-ss)
>   f = fix(x);
>   p = fix((x-f) .* 60);
>   m = ((x-f).*60-p).*60;
>   str = sprintf('%4d-%02d-%02.0f', f, abs(p), abs(m));
> end
```

Próbáljuk ki! (Az abszolút érték a negatív szögek jó megjelenítéséhez kell.)

```
> x = 123.456789, y = -23.00987
> fpm(x), fpm(y)
```

LOGIKAI MŰVELETEK

A logikai műveletek (1-igaz/0-hamis) ismerete is nagyon fontos, különösen, ami a vektorok/mátrixok elemeinek módosítását, lekérdezését illeti. Sok olyan feladatot is meg tudunk oldani a mátrixokkal és logikai változókkal, amihez különben valamilyen ciklus kellene.

```
> % egyenlo ==, nem egyenlo ~=
> b = 3==4
> whos b
> b = 5~=6
> vs = [1 2 3 4 5 6] % sorvektor
> vs(5)>5
> vs(5)>=5
> % vagy (or) ||, es (and) &&
> vs(1)>2 || vs(4)>2 % igaz, mert a ket feltetel kozott van igaz
> vs(1)>2 && vs(4)>2 % hamis, mert csak az egyik feltetel igaz
```

Nézzünk egy példát, ahol egy vektor adott tulajdonságú elemeit kérdezzük le logikai változóval. Legyen egy műegyetemi tanár, aki véletlenszerűen osztogatja a jegyeket a diákoknak a vizsgán. Most éppen 6 vizsgázója van (a,b,c,d,e,f). Kapjon mindenki egy jegyet 1-5 között és utána kérdezzük le hányan buktak és kik?

```
> vizsgazok = ['a';'b';'c';'d';'e';'f']
> vizsga = ceil(rand(1,6)*5)
> bukott = vizsga<2
> vizsgazok(bukott)
```

A bukott=vizsga<2 eredménye egy 10 elemű vektor lesz, ahol 1-es áll azokon a helyeken, amire igaz volt a feltétel, 0 a többin. Ha a vizsgázók nevei közül le akarjuk kérdezni azokat, akik megbuktak, akkor nem kell mást tennünk, mint meghívni a vizsgazok(bukott) parancsot, ez csak azokat a neveket fogja visszaadni, ahol 1-es állt a bukott vektorban. Egy ilyen lekérdezést matlab-ban ciklus nélkül is meg lehet oldani, logikai változókat használva.

Megjegyzés: kerekítési parancsok: round, ceil, floor, fix (lásd help)

CIKLUSOK

Persze azért mindent nem lehet ciklusok használata nélkül megoldani, úgyhogy ismerkedjünk meg kicsit a ciklusokkal! Kétféle ciklust használunk, számlálással és feltétellel vezérelt ciklust.

SZÁMLÁLÁSSAL VEZÉRELT CIKLUS

Írjunk egy programot, ami kirajzolja x^1, x^2, x^3, x^4, x^5 függvényt [-10, 10] intervallumon és el is menti ezeket képként! Először csak plottoljuk ki x^2 függvényt és mentsük el!

```
> x = -10:0.1:10
> y = x.^2
> plot(x,y)
> print('proba.jpg','-djpeg')
```

Nézzük meg a megoldást számlálással vezérelt for ciklust használva!

```
> for i=1:5
>   y = x.^i;
>   plot(x,y);
>   fajlnev = sprintf('fuggveny%04d.jpg',i)
>   print(fajlnev,'-djpeg')
> end
```

FELTÉTELLEL VEZÉRELT CIKLUS

Egy másik műegyetemi oktató kicsit tudományosabban adja a jegyeket a vizsgáján, nem egyenletes eloszlásban, hanem normális eloszlásban, 3-as várható értékkel és 1-es szórással. Vajon hányszor kell elmenjek hozzá vizsgázni, ha 5-öst szeretnék kapni? Írjunk egy programot, ami előállít egy véletlen számot először egy for ciklusban 100-szor, 3-as várható értékkel, 1-es szórással, egészre kerekítve. Előfordulhat, hogy 1-esnél rosszabb vagy 5-ösnél jobb jegyet kapunk (nézzük meg a min, max értékét!). Ezeket logikai változók használatával módosítsuk 1-esre és 5-ösre. Ábrázoljuk hisztogramon! Majd írjunk egy feltétel vezérelt ciklust, ami a saját vizsgáinkra vonatkozik, ami akkor áll le, ha végre 5-öst kaptunk!

Először nézzük meg egy for ciklusban 100-szor az osztályozást!

```
> R = [];
> for i=1:100
>   r = round(randn(1)+3);
>   R = [R r];
> end
> R
> max(R), min(R)
> R(R<1)=1;R(R>5)=5
> hist(R)
```

Most a saját vizsgáinkat nézzük, addig fusson a ciklus, amíg 5-öst (vagy jobbat) nem kapunk! Hányadikra sikerült?

```
> r = 0; i =0; R = [];
> while r<5
>   r = round(randn(1)+3);
>   R = [R r];
>   i = i+1;
> end
> R
> i
```

Ha szükséges a `break` utasítással ki lehet lépni pl. bizonyos feltétel teljesülése esetén mind a `for`, mind a `while` ciklusból. A `continue` parancs adott feltétel esetén kihagyja a ciklus további részét és a következő iterációs lépésre ugrik.

ELÁGAZÁSOK

Írjunk saját függvényt a második geodéziai főfeladat meghatározására, vagyis két koordináta pár alapján távolság és irányszög meghatározására. Vegyük figyelembe a különböző síknegyedeket is!

A képletek: $t = \sqrt{\Delta x^2 + \Delta y^2}$; $\alpha = arctan\left|\frac{\Delta y}{\Delta x}\right|$

Síknegyedenként: 1. $\delta = \alpha$, 2. $\delta = 180° - \alpha$, 3. . $\delta = 180° + \alpha$, 4. $\delta = 360° - \alpha$

```
> function [t delta] = tav_irany(y1,x1,y2,x2)
>                  dy = y2 - y1;
>                  dx = x2 - x1;
>                  t = sqrt(dy^2+dx^2);
>                  alfa = atand(abs(dy/dx));
>                  if dy>=0 && dx>=0
>                       delta = alfa;
>                  elseif dy>=0 && dx<0
>                       delta = 180 - alfa;
>                  elseif dy<0 && dx<0
>                       delta = 180 + alfa;
>                  else
>                       delta = 360 - alfa;
>                  end
> end
```

Ellenőrizzük le parancssorból a 4 negyedet!

```
> tav_irany(0,0,10,10)
> tav_irany(0,0,10,-10)
> tav_irany(0,0,-10,-10)
> tav_irany(0,0,-10,10)
```

Egyszerűbb a megoldás, ha ismerjük az `atan2` (eredménye radian) illetve `atan2d` (eredménye fok) parancsot! Ez síknegyedekre lebontva számolja a szögeket, az utóbbi -180 és +180 fok között. Itt csupán 360 fokot hozzá kell adni a negatív értékekhez.

```
> function [t delta] = tav_irany2(y1,x1,y2,x2)
>                  dy = y2 - y1;
>                  dx = x2 - x1;
>                  t = sqrt(dy^2+dx^2);
>                  delta = atan2d(dy,dx); % Octave esetén
>                  if delta<0 delta=delta+360; end
> end;
```

Az elágazások másik alakja a switch-case utasítások.

```
> muszer = menu('Milyen muszert kersz?','Szintezo', 'Teodolit', 'GPS');
> switch muszer
>   case 1
>     disp('Szintezo kiadva!')
>   case 2
>     disp('Teodolit kiadva!')
>   case 3
>     disp('GPS kiadva!')
> end
```

TRY - CATCH ALGORITMUS

Sokszor előfordul, hogy bizonyos műveletek során hiba lép fel. Alapesetben erre leáll a program futása. Ha a hiba ellenére is szeretnénk tovább futtatni a programot, akkor használhatjuk a try-catch algoritmus, ilyenkor megadhatjuk, hogy hiba esetén (pl. hiányzó fájl, amit be akarunk tölteni, hibás művelet) mi történjen. Próbáljuk ki a következőt!

```
> A = rand(5,3)
> B = rand(5,4)
> C = [A; B]
> D = [A B]
```

Hibaüzenettel leállt a program a C után. Az alábbi esetben viszont csak egy figyelmeztető üzenetet ír ki, de fut tovább.

```
> try
> C = [A; B]
> catch
> disp('Nem egyeznek a matrixok meretei')
> end
> D = [A B]
> disp('Itt a vege!')
```