8–9. GYAKORLAT

A MATLAB PROGRAMOZÁSA

BEVEZETÉS

Eml. (ea.): A Matlab beépített programozási nyelve elemként tartalmazza mindazon vezérlőszerkezeteket, amelyek a strukturált programok kialakításánál használhatók. Tantárgyunk keretén belül mi strukturált programokat készítünk, tehát a szekvencia, továbbá szelekciós és iterációs szerkezetek fordulnak majd elő a programjainkban (goto-szerű szerkezeteket nem fogunk használni).
A legegyszerűbb esetektől (1-2 soros kód) eltekintve a programkódot nem a parancsablakba írjuk, hanem script fájlokban vagy saját függvényekben helyezzük el.
Feladataink lassan (átlagosan) egyre nehezednek, egy-egy blokkon belül és a blokkok között is. Senkit ne tántorítson el, ha kezdetben egy-egy nehezebb feladat megoldása túl komoly kihívásnak bizonyul. Gyakorlással és a programozási tapasztalat megszerzésével érzékelhető lesz a fejlődés!
Megjegyezzük, hogy az óra célja alapvetően a gyakoroltatás (programozás tanítás), tehát annak ellenére, hogy a Matlabba beépítve – általában – léteznek fejlett eszközök az általunk kitűzött problémák megoldására, ezeket sokszor *szándékosan* nem fogjuk most használni. A fejlett matlabos megvalósítások kipróbálása (a teljes lehetséges eszközkészlettel) szorgalmi házi feladatnak marad.

Feladat
Idézzük fel, hogy mit tanultunk a saját függvényekről (és a script m-fájlokról)! Egy egyszerű példán próbáljuk ki (újra) a paraméterek szerepét (formális és aktuális paraméterek), ill. gondoljuk át (újra), hogy a [] és () zárójelek mikor hagyhatók el a fejlécből.
Kirajzoltathatunk például egy szinusz függvényt, vagy kiírhatjuk egy szám négyzetét.

SZEKVENCIA

Feladat
Idézzük fel, hogy hogyan valósítja meg a Matlab a szekvenciát! Próbáljuk ki egyszerű példákon (parancsablak; akár több sorba is írhatunk). Mi a vessző és a pontosvessző szerepe?
Egyszerű (csak szekvenciát tartalmazó) scriptet is írhatunk, pl. kör kerületének és területének számolására, vagy mágikus mátrix kiíratására. (A sugár, ill. a méret megadásának lehetőségeit lásd lent.)

A Matlab környezet fontos jellemzője (eltérően a legtöbb programozási nyelvtől/környezettől, pl. C, VBA) hogy itt *nincs változódeklaráció*, az adott változó (objektum) típusa értékadáskor dől el (persze ez később módosítható).

Feladat
Idézzük fel, hogy mit tanultunk korábban a különböző típusú adatok kezeléséről (numerikus és szöveges típusok). Térjünk ki a következőkre: értékadás (határok), hasonlítások, műveletek, konverziók. Mutassunk be példákat és néhány jellegzetes hibát is!

Feladat
Készítsünk megfelelően felparaméterezett saját függvényt (vagy scriptet) a következő probléma megoldására. (A bekérés jelentheti paraméterek megadását az eljárásban, ill. használhatjuk az Input függvényt is. A kiírást intézhetjük egyszerűen, vagy szebben is.)
Egy sztringben egy olyan név szerepel, amely két részből áll (családnév, keresztnév), a két részt pontosan egy szóköz választja el. Cseréljük fel a név két részét!
(Tipp: Használjuk a *findstr* függvényt (+ esetleg az *strcat* és *length* függvényt is)! Célszerű létrehozni külön a két névrészt, és utána összeilleszteni.)
A megoldás vázlata:

```
>> st = 'Makk Marci'
>> szokoz_poz = findstr(' ',st)    % A válasz 5 lesz
>> st1 = st(1: szokoz_poz-1)       % Első sztringfél
>> st2 = st(szokoz_poz+1:end)
```

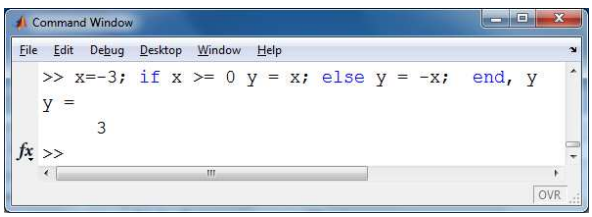
Feladat
Készítsünk megfelelően felparaméterezett saját függvényt (vagy scriptet) a következő probléma megoldására. (Bekérés mint előbb.)
Adott egy síkbeli háromszög három pontja, *x* és *y* koordinátákkal. Határozzuk meg a háromszög kerületét; területét és szögeit.
Tipp: A kerület a pontok távolságaiból adódik. A területre használjuk Héron képletét (1). A háromszög szögei a terület ismeretében számíthatók a (2) képletből (értelemszerűen aktualizálva a többi szögre).

Képletek: (1) $t = \sqrt{s(s-a)(s-b)(s-c)}$, (2) $t = \frac{a \cdot b \cdot \sin\gamma}{2}$.

SZELEKCIÓ

Idézzük fel (előadás), ill. tanulmányozzuk a súgót, hogy hogyan valósítja meg a Matlab a „sima” szelekciót és az elseif-es szerkezetet! Jegyezzük meg a szintaktikát!

Feladat
Próbáljuk ki (teszteljük) az előadáson bemutatott „absval” és „signum” (saját) függvényeket! Módosítsuk a függvényeket úgy, hogy a szeparátorjeleket módosítjuk, elhagyjuk (ahol lehet). Futtassuk a függvényeket úgy is, hogy nemcsak a javasolt típusú paraméterrel hívjuk őket. Mit tapasztalunk?
A kód mindkét esetben a parancsablakba is beírható, próbáljuk ki ezt is!



Feladatok

Készítsünk megfelelően felparaméterezett saját függvényt (vagy scriptet) a következő problémák megoldására. (A bekérés jelentheti paraméterek megadását az eljárásban, ill. használhatjuk az Input függvényt is. A kiírást intézhetjük egyszerűen, vagy szebben is.)

1. Kérjünk be két egész (valós) számot és írjuk ki a nagyobbik szám értékét!

2. Döntsük el, hogy egy generált véletlenszám kisebb vagy nem kisebb 0,5-nél!
(Tipp: Az ismert *rand* mellett használjuk a *round* függvényt – ez a legközelebbi egészre kerekít.)

3. Készítsünk programot, amely bekér a felhasználótól egy fizetést, és a fizetés nagyságától függően kiírja, hogy a jövedelem alacsony, átlagos, vagy magas! A kategóriákat mindenki a saját preferenciája alapján rögzítheti. ☺

4. Adott (bekért) három szám esetén mondjuk meg azt, hogy lehet-e a három szám egy síkbeli háromszög három oldalának a hossza vagy sem!
(Tipp: Ha bármelyik két oldal hosszának összege nagyobb, mint a harmadik oldal hossza, akkor „lehet”, egyébként „nem”.)

5. Adott két kör a síkon. Mindkét kör a középpontjának koordinátáival és a sugarával adott. Mondjuk meg azt, hogy van-e a köröknek közös pontja vagy sem!
(Tipp: Ha a középpontok (Pitagorasz tétellel kiszámolt) távolsága nagyobb, mint a két sugár összege, akkor „nincs”, egyébként „van”.)
Fejlesszük tovább a programot azzal, hogy a közös pontok darabszámát is kiírjuk!

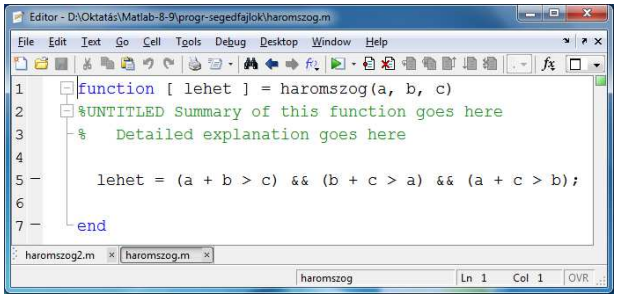
Egy lehetséges megoldás a 2. feladathoz (szép kiírással):

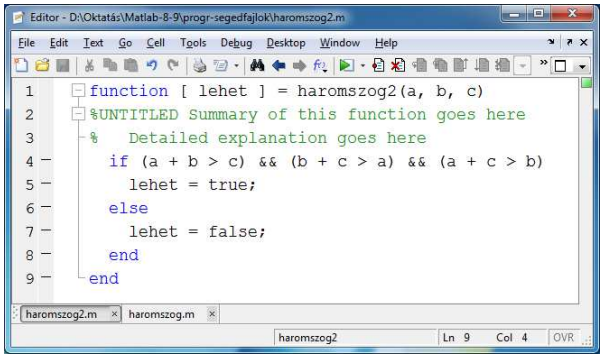
```
>> t=rand(); if round(t) s=' nem kisebb 0.5-nél'; else s=' kisebb, mint 0.5'; end;
>> strcat(mat2str(t,5), s)
```

A 3. feladat egy lehetséges egyszerű megoldása (fizetés 1000 Ft-ban):

```
>> fiz=200;
>> if fiz<150 disp('alacsony'), elseif fiz<270 disp('átlagos'), else disp('magas'), end
```

A negyedik feladat lehetséges megoldásai:





Feladat
Mi lesz az eredménye a következő kódrészletnek (és miért)?

```
>> if eye(3) a=1, else a=0, end, a
```

Értékeljünk ki további hasonló logikai feltételeket, amelyekben vektorok, ill. mátrixok szerepelnek!

ITERÁCIÓ

Tanulmányozzuk a súgót, hogy hogyan valósítja meg a Matlab a *for* ciklust és a *while* ciklust! Jegyezzük meg a szintaktikát!

Feladatok
Egyszerű példákkal próbáljuk ki a ciklusok működését! A *for* ciklusnál például számolhatjuk pozitív egész számok összegét vagy szorzatát (faktoriális), a *while* ciklusnál pedig használhatjuk a gépi epszilonos példát vagy annak egy módosítását. Utóbbi esetben figyeljünk a feltétel gondos megadására! Kiírást is rakjunk a ciklusaink végére!
A kezdetben üres *m* sorvektort töltsük fel páratlan számokkal 1-től 11-ig, for ciklussal!
Hasonlóan, az *n* sorvektorban helyezzük el az első 11 hárommal osztható pozitív egész számot!
Írjuk ki az egész számokat 1-10-ig *a két különböző ciklusszervezéssel*!

Feladat
Mit csinál a következő kódrészlet?

```
>> A = zeros(4)
>> for i = 1:4, for j = 1:4, A(i,j)=(i-1)*length(A)+j; end, end
```

Írjuk át a kódot úgy, hogy ne kelljen for ciklusokat használni! (Ez már egy egyszerű vektorizációs példa, bővebben lásd lent.)

Feladat
Írjuk át a Hilbert-mátrix előállítására készített kódot (két egymásba ágyazott *for* ciklus) úgy, hogy a konstrukció saját függvénnyel legyen meghívható, és paraméterként lehessen megadni a mátrix méretét!
Készítsünk olyan – szintén *n*-nel paraméterezett – saját Hilbert-függvényt is, amely nem használ *for* ciklusokat az eredmény előállítására!

Feladatok
Készítsünk megfelelően felparaméterezett saját függvényt (vagy scriptet) a következő problémák megoldására. Több feladatra létezik a Matlabban beépített függvény, ill. eszköz, de ne azt használjuk. (Bekérés és kiírás mint korábban.)

1. Írjunk programot, amely kiírja az első *n* darab négyzetszámot!
Írjuk ki a képernyőre az összes *n*-nél kisebb négyzetszámot! (Figyelem, ez más feladat, mint az előző!)

2. Adott *n* pozitív egész számra írjuk ki a Fibonacci sorozat első *n* elemét! A képzés módszere: az első két elem 1, ezután minden következő az előző kettő összege (Pl. *n* = 6: 1, 1, 2, 3, 5, 8). (Tipp: Az $n \leq 2$ esetek külön kezelendők, egyébként meg használjunk három változót és egy ciklust a feladat megoldására!)

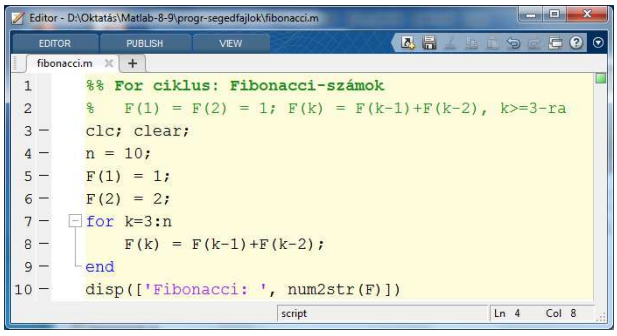
3. Írjunk programot, amely kiírja a 4, 5 és 6 jegyekből képezhető összes négyjegyű számot! (Tipp: Minden pozíción a jegyek előállítására használjunk egy-egy for ciklust.)

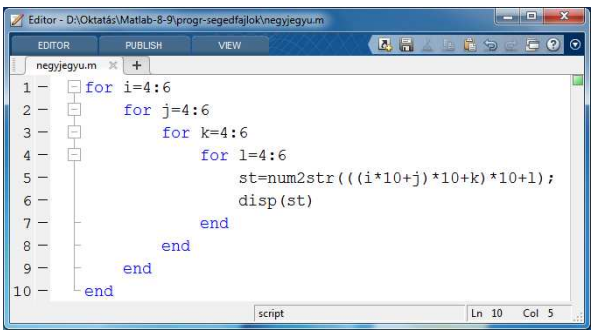
4. Írjuk ki egy szám összes osztóját (1-et és önmagát is)! (Tipp: lásd a megjegyzést a *mod* függvényről a következő feladatnál.)

5. Határozzuk meg két pozitív egész szám legkisebb közös többszörösét! (Tipp: Jelölje a kisebbik számot *a*, a másikat *b*. Vizsgáljuk *a* többszöröseit (*a*, 2·*a*, 3·*a*, …) egészen addig, amíg *b* egy többszörösét nem kapjuk. Ilyen szám előbb utóbb lesz, ha más nem, akkor *a*·*b*. Egy *c* szám *b* többszöröse, ha *b* megvan benne maradék nélkül, azaz *c* mod *b* = 0. A Matlabban is létezik *mod* függvény.)

6. Döntsük el két pozitív egész számról, hogy relatív prímek-e vagy sem! Két szám relatív prím, ha 1-en kívül nincs más közös osztójuk. (Tipp: Vizsgáljuk meg az egész számokat kettőtől kezdve (2, 3, …) legfeljebb a kisebbik számig! Ha a vizsgált szám mindkét számot osztja (azaz megvan bennük maradék nélkül), akkor a két szám „nem relatív prím”, egyébként, ha nem találtunk a kisebbik számig bezárólag ilyen számot, akkor a két szám „relatív prím”.)

A 2. és a 3. feladat egy-egy lehetséges megoldása:





INPUT ÉS OUTPUT

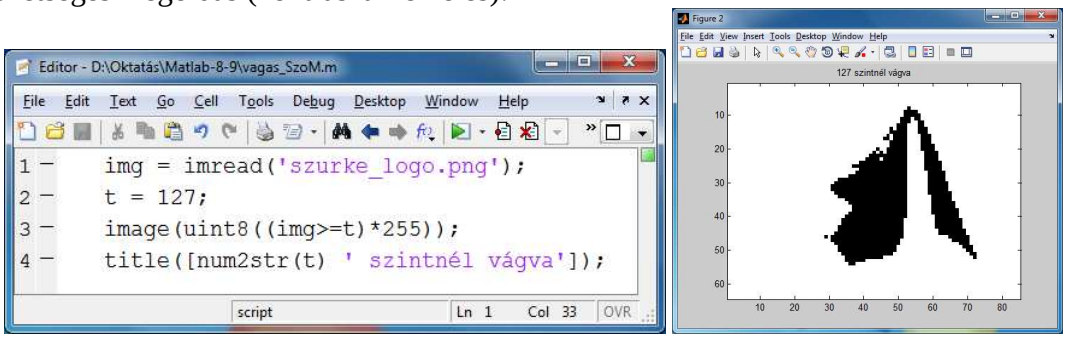
Idézzük fel, hogy mi hangzott el az előadáson a „szép” bekérésről és kiírásról. Ha szükséges, tájékozódjunk a súgóban a megismert függvényekről.

Feladat
Alakítsuk át fenti függvényeinket, eljárásainkat úgy, hogy az adatokat szépen kérjék be, és szépen írják ki. (Pl. háromszög oldalainak bekérése stb.)

A KÓD HATÉKONYSÁGÁNAK NÖVELÉSE

Idézzük fel az előadás fóliákon bemutatott technikákat a kód hatékonyságának növeléséről! Próbáljuk ki az ottani példákat, és készítsünk saját magunk is hasonlókat! Mérjük le a végrehajtási időket!

Feladat*
Írjuk át a képszerkesztés óra néhány feladatát úgy, hogy „for” ciklus(ok) helyett vektorizált módon kezelje a tömböket, ill. vektorokat!
Egy lehetséges megoldás (kontraszt kiemelés):



OTTHONI MUNKA

Feladat (szekvencia)
Készítsünk programot, amelyben a felhasználó megadja egy tetszőleges négyszög csúcsainak koordinátáit, a program pedig kiszámítja a négyszög kerületét. Feltételezhetjük, hogy a bevitt pontok által meghatározott szakaszok nem metszik egymást (a programban nem kell ellenőrizni). Számítsuk ki a területet is! (Tipp: Ehhez Hérón képlete használható, úgy, hogy a négyszöget két háromszögre vágjuk szét.)

Feladat (szelekció)
Oldjuk meg (saját scripttel) az $a \cdot x^2 + b \cdot x + c = 0$ másodfokú egyenletet a valós számok körében, ahol az *a*, *b*, és *c* együtthatók értékei valós számok!
(Tipp: A feladat teljes megoldása kezeli az elsőfokú eseteket és a másodfokú eseteket (0, 1, ill. 2 db valós gyök) is! Most ne használjuk a Matlab megoldó eszközeit.)

Feladat (szelekció)
Készítsünk programot, amely képes megoldani a 2×2-es lineáris egyenletrendszert!
Tipp: Legyenek a paraméterek rendre *a*, *b*, *c*, *d*, *e* és *f*, az ismeretlenek pedig *x* és *y*. Ha megoldás egyértelmű, adjuk meg, ha végtelen sok van, válasszunk egy tetszőlegeset. Ellentmondásos esetben egyszerűen írjuk ki, hogy nincs megoldás.

Feladat (iteráció)
Adott *n* és *k* pozitív egész számokra határozzuk meg a binomiális együttható értékét!

(Tipp: A képletet (n!/(k!*(n − k)!)) ne a számláló és a nevező kiszámolása utáni osztással számoljuk ki (a nagy számok miatti pontatlanság/túlcsordulás miatt), hanem előbb egyszerűsítsük a képletet k!-sal, majd a kapott törtet bontsuk n − k db tört szorzatára és ezt számoljuk ki!)

Feladat
Írjunk eljárást az Euklideszi algoritmus megvalósítására (legnagyobb közös osztó megkeresése ismételt maradékos osztásokkal)! Használjuk a jegyzetben bemutatott kódrészletet!

```
>> while q>0 r = mod(p,q); p = q; q = r; end; lnko = p
```

Feladat*
Töltsünk fel egy 1×20-as méretű vektort egész számokkal (használhatunk véletlen egészeket). Írjuk meg a minimum kiválasztásos és a buborékos rendező eljárásokat Matlabban, és rendezzük a vektorunkat!

Feladat*
Készítsünk rutint A*x = b típusú egyenletrendszerek megoldására! (Először a négyzetes esetet „fedjük le”, majd próbáljuk még általánosítani a megoldást!)

Feladat (verseny)*
Adjunk megoldást Matlabban az Euler projekt feladataira! (Az első néhányra.)
(https://projecteuler.net)