



Matlab 8–9. előadás

A Matlab programozása

Dr. Szörényi Miklós,
Dr. Kallós Gábor

2017–2018





Tartalom

- Algoritmus, pseudokód, program
 - Programozási alapfeladatok
- Strukturált programozás
 - Vezérlő szerkezetek
- Elágazások
 - Két- és többágú szelekció
 - Feltételek, kiértékelés
- Ciklusok
 - For ciklus
 - Egymásba ágyazott ciklusok
 - While ciklus
- Rekurzió
- Bekérés és kiírás
 - Interaktív grafika
- Paraméterek kezelése
- A kód hatékonysága





Algoritmikus feladatmegoldás

Fontos fogalmak

- Számítási folyamat: a számítógépben zajlik
 - Adatmanipuláció (egy cél érdekében)
 - A működést *programok* vezérlik
- Szeretnénk mi is *programot készíteni*
- *Specifikáció*: kapunk egy feladatot, ill. egy leírást a feladatról
 - Mit kell csinálnia a programnak?
 - Mi lesz a bemenet, és mi lesz a kimenet
- *Algoritmus*: hogyan oldja meg a feladatot a program?
 - A megoldás *lépésről-lépésre* történő leírása
 - Nekünk kell kitalálni, nincs rá teljesen általános módszer
 - Az algoritmus kidolgozása egyszerre tudomány, „művészet” és mérnöki feladat
 - Pl. helyesség-bizonyítás, tulajdonságok számszerűsíthetőek, de: kellenek egyedi ötletek, felismerések, ugyanakkor vannak *szabványok*





Algoritmikus feladatmegoldás

Fontos fogalmak

- Algoritmus
 - Utasítássorozat megengedett lépésekből
 - Mechanikusan (gépiesen) végrehajtható
 - Jó esetben nem kell emberi beavatkozás
 - *Véges sok lépésből áll, mindig egyértelműen adott a következő lépés*
 - Minden időpontban *véges sok* memória kell (tehát nem végtelen)
 - A használt adatokat (változókat) az alg. tervezése előtt/közben össze kell gyűjteni!
- Egyszerű algoritmus példák
 - Számsorozat összeadása
 - Prímtényezős felbontás
- *Pszeudokód*: eszköz az algoritmus szöveges leírására
 - Mindig egy adott formalizmust követ (ez lehet „nagyvonalú”)
- *Program(kód)*: a szg. által érthető nyelven, megfelelő szabályokat betartva (szigorúan formálisan!) leírt algoritmus





Algoritmikus feladatmegoldás

Fontos fogalmak

- **Változó:** eltárolt, *megjegyzett érték*, amelynek nevet adunk
 - Vagy: egy hely neve (a memóriában), ahol egy értéket tárolunk
 - A változót lehet (ki)olvasni és lehet neki *értéket adni* (az érték tehát változhat)
 - Értékadáskor a régi érték elveszik
 - A program futása során, egyes időpontokban más-más lehet egy adott változó értéke
 - Ugyanaz a művelet máskor más eredményt adhat
- **Konstans:** egy helyre (a memóriában) fixen beírt érték
 - Értéke *nem változhat*
- **Típus:** értékkészlet és műveletek együttese
 - A változó típusa egyben a vele végezhető műveleteket is meghatározza
 - Klasszikus (egyszerű) típusok: egész, valós, logikai, karakter, szöveg
 - Pl. a logikai típusú változó ÉK-e: {igaz, hamis}, műveletek: „nem”, „és”, „vagy” stb.
 - Klasszikus összetett típusok: tömb (vektor, mátrix), rekord, halmaz stb.





Algoritmikus feladatmegoldás

Fontos fogalmak

■ Kifejezések

- Változókból, konstansokból és műveletekből (ill. relációkból) épülnek fel
- Meg kell, hogy feleljenek a nyelvtani szabályoknak
- A műveletek adott típusokon végezhetőek el, és az eredménynek is van valamilyen típusa (utóbbi lehet esetleg más)
 - Pl. egy szám+szám művelet eredménye is szám, de egy szám<szám összehasonlítás logikai eredményt ad

■ Deklaráció

- A változókat sok programozási nyelvben (előre) *deklarálni kell*
- Erősen típusos programozási nyelvek: minden változót definiálni kell, előre meg kell mondani a típusát és a nevét (pl. C nyelv)
- Gyengén típusos nyelvek: a változóknak nem kell előre megadni a típusát, sőt egy adott nevű változó típusa is változhat (pl. Matlab)





Strukturált programozás

- A program megengedett elemei:
- Elemi műveletek
 - Kiírás, bekérés
 - Értékadás, valaminek a kiszámolása
- Szekvencia
 - Egymás utáni lépések (a feladat részekre bontható)
 - Számít a sorrend!
- Szelekció (elágazás)
 - Feltételes végrehajtás
 - Logikai kifejezés vizsgálata, igaz ág, hamis ág
 - Minden futtatásnál csak egy konkrét ág hajtódik végre
- Iteráció (ciklus)
 - Programrész ismétlése, amíg egy feltétel fennáll
 - Ciklusmag, ciklusfeltétel (mindig újra kiértékelődik), c.fej, c.változó
 - Elöl- és hátultesztelő ciklus
- *Nem megengedett* például az ugrás (goto) vagy a kiugrás (break)
- Matematikailag bizonyított: minden számítógéppel megoldható feladat megoldható *strukturáltan is*
- Célszerű a kódot megfelelően tördelni
 - Egymásba ágyazott szerkezetek





Programozási alapeladatok

- Tipikusan egy (adott/bekért) sorozat elemeit dolgozzuk fel
 - A sorozat elemszáma ismert lehet, vagy a végét spec. jel jelzi (pl. -1)
 - Ettől függően for ciklus vagy while ciklus használandó
 - A feladatok egyszerűbb esetben tömbök nélkül (is) megoldhatók
- Összegzés
 - Példa: számok összeadása, faktoriális
 - Összeg vagy szorzat változó használandó
- Számlálás
 - Példa: osztók száma, páros számok száma, adott karakterek száma
 - Darab változó használandó
- Szélsőértékek keresése
 - Példa: maximum- és minimumkeresés
 - Min/max változó használandó
 - Az első elem célszerűen külön kezelendő
- Eldöntési feladatok; szerepel-e egy adott érték a sorozatban
 - Példa: n prímszám-e
 - Logikai változó használandó
- Keresés, rendezés





Áttekintés

A Matlab programozási környezetéről

- Saját programozási nyelv
 - Felhasználóbarát, de alacsonyszintű elemek is vannak/használhatók
- Barátságos fejlesztőkörnyezet
 - Szintaktikai kiemelések, fordítási ellenőrzés, javítási javaslatok stb.
- Változók, kifejezések
 - Nem deklaráljuk a változókat
 - Típusok, értékhatárok, műveletek, pontosság, konverziók stb. – tudjuk
 - Logikai műveletek, hasonlítások az ismert módon
- A strukturált programozás elveit követjük
 - Építőelemek: szekvencia, szelekció, iteráció
 - A végrehajtás alapvetően szekvenciális, de párhuzamos szerkezetek is beépíthetők
 - A vezérlőszerkezeteknek van eleje és vége (utóbbi tipikusan: end)
- Célszerűen m-fájlokban tároljuk a kódot
 - Tipikusan nem a parancsablakban dolgozunk
- Hibakeresési lehetőségek





Vezérlő szerkezetek/Szekvencia

- *Szekvencia*: műveletek egymás utáni végrehajtása (az eredeti feladat felbontásával)
 - Elemi művelet: tovább már nem bontható (tudjuk)
- A csak szekvenciát tartalmazó prg.ok a legegyszerűbbek (általában)
- Szekvencia példa: kör kerülete és területe
 - Feladat: Írjuk át bekérősrre (input parancs), ill. bővítsük szép kiírással (*disp* parancs)
- Másik hasonló példa: mágikus mátrix kiírása (script, gyak.)

```
Editor - D:\Oktatás\Matlab-4-5\kor.m
EDITOR PUBLISH VIEW
kor.m x +
1 function [ t, k ] = kor( r )
2 % Kör területének és kerületének kiszámítása
3 % Adott a sugár
4 % Hívás pl.: [ter ker] = kor(1)
5 t = r.^2*pi;
6 k = 2*r*pi;
7 end
kor Ln 6 Col 14
```

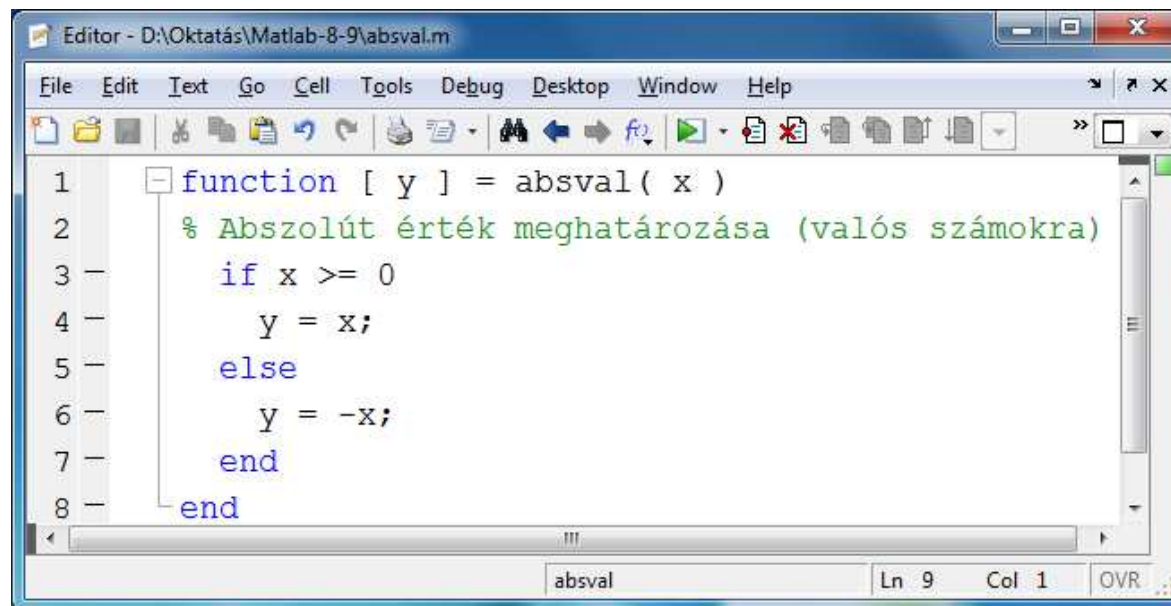




Vezérlő szerkezetek/Elágazások

„Sima” szelekció

- Motiváció: A „fix” viselkedéstől eltérően, helyzettől függően a program különböző utasításokat hajt(hat) végre
- A klasszikus if-then-else szerkezet szintaktikája:
`if` logikai kifejezés[,|;] utasítás(ok), `else` utasítás(ok),
`end`
 - Megj.: A Matlabban nem használjuk a `then` kulcsszót!
- Példa: Abszolút érték számítása (saját függvény)
 - Elemezzük a működést! (Teszteljük a fv-t, hasonlítsuk össze a beépített változattal)
 - Mi történik komplex input esetén?



```
Editor - D:\Oktatás\Matlab-8-9\absval.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [ y ] = absval( x )
2 % Abszolút érték meghatározása (valós számokra)
3 if x >= 0
4     y = x;
5 else
6     y = -x;
7 end
8 end
absval Ln 9 Col 1 OVR
```



Elágazások

Elseif-es szelekció

- Újabb ág beépítése szükséges (több mint két alternatíva)
 - Egymást kizáró ágak
 - *else*-be ágyazott újabb *if*-fel is kiváltható, de az *elseif* általában elegánsabb
- Szintaktika értelemszerűen
- Példa: Előjel számítása (saját függvény)
 - Elemezzük a működést! (Teszteljük a fv-t, hasonlítsuk össze a beépített változattal – *sign*)

```
Editor - D:\Oktatás\Matlab-8-9\progr-segedfajlok\signum.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [ y ] = signum( x )
2 % Előjel meghatározása (valós számokra)
3 if x > 0
4     y = 1;
5 elseif x == 0
6     y = 0;
7 else
8     y = -1;
9 end
10 end
```

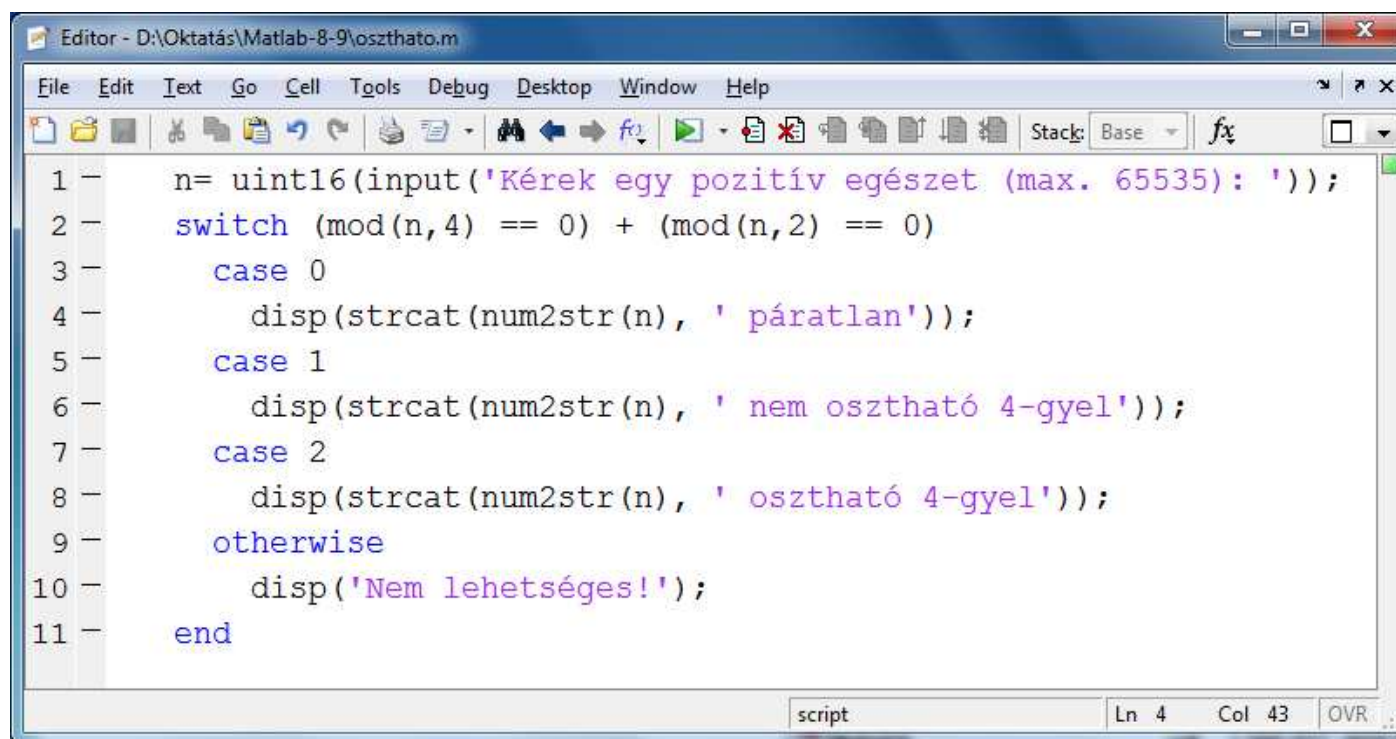




Elágazások

Switch szerkezet

- Többfelé elágazást tesz lehetővé, egy kifejezés értékétől függően
 - Általában több mint két alternatíva
- Szintaktika: lásd Súlyó
 - *switch*, *case* ágak, *otherwise*, a végén *end*
- Példa: Oszthatósági vizsgálat (m-fájl, script)
 - Az *strcat* sztringek összefűzését hajtja végre



```
Editor - D:\Oktatás\Matlab-8-9\oszhato.m
File Edit Text Go Cell Tools Debug Desktop Window Help
[Icons] Stack: Base fx
1 - n= uint16(input('Kérek egy pozitív egészet (max. 65535): '));
2 - switch (mod(n,4) == 0) + (mod(n,2) == 0)
3 -     case 0
4 -         disp(strcat(num2str(n), ' páratlan'));
5 -     case 1
6 -         disp(strcat(num2str(n), ' nem osztható 4-gyel'));
7 -     case 2
8 -         disp(strcat(num2str(n), ' osztható 4-gyel'));
9 -     otherwise
10 -         disp('Nem lehetséges!');
11 - end
script Ln 4 Col 43 OVR
```



Logikai kifejezések

- Feltételek kiértékelésénél
- Felépítés
 - Relációk (tudjuk már): $<$, $>$, $\leq$, $==$, $\sim =$ stb.
 - (Relációs operátorok: *eq*, *gt*, *le* stb.)
 - Logikai operátorok (tudjuk már): $|$ (*or*), $\&$ (*and*), $\sim$ (*not*), *xor*, $||$, $\&\&$ (+*any*, *all*)
 - Egyes adatkezelő függvények: *strcmp*, *length*, *size*, *isempty*, *isnumeric*, *isinf*, *isfinite*, *isequal*, *isstr*, *ismember*, *upper*, *lower* stb. (lásd később is)
- Kiértékelés
 - Zárójelezés, precedencia, balról-jobbra (tudjuk)
 - Specialitás: rövidzár (sokszor elég az első op. kiértékelése)
 - A feltétel tömbértékű kifejezés is lehet, ha van benne 0, hamisnak minősül (pl. *eye(3)* logikai 0, *ones(3)* logikai 1)
- Példák
 - `if isnumeric(b) || (a > b)`
 - `if ismember(a, set)`
 - `while (b ~= 0) && (a/b > 18.5)`
 - `if exist('myfun.m') && (myfun(x) >= y)`
 - `elseif abs(m - n) == 2`





Vezérlő szerkezetek/Ciklusok

For ciklus

- A legegyszerűbb iteráció (fix lépésszám)
- Motiváció: Egy parancsot vagy parancsok egy csoportját néhányszor/többször meg kell ismételni
- Szintaktika:
`for ciklusvált = kezdő:végé[, |;] utasítás(ok), end`
 - A ciklusváltozót nemcsak egyesével léptethetjük, pl. `i = 1:2:6`
- Példa: Faktoriális (saját függvény)
 - Elemezzük a működést! (Teszteljük a fv-t különböző típusú adatokkal is)

```
Editor - D:\Oktatás\Matlab-8-9\fakt.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Base fx
1 function [ f ] = fakt( n )
2 % Faktoriális meghatározása (adott poz. egész értékig)
3 f = 1;
4 for i = 2:n
5     f = f*i;
6 end
7 end
fakt Ln 8 Col 1 OVR
```



Ciklusok

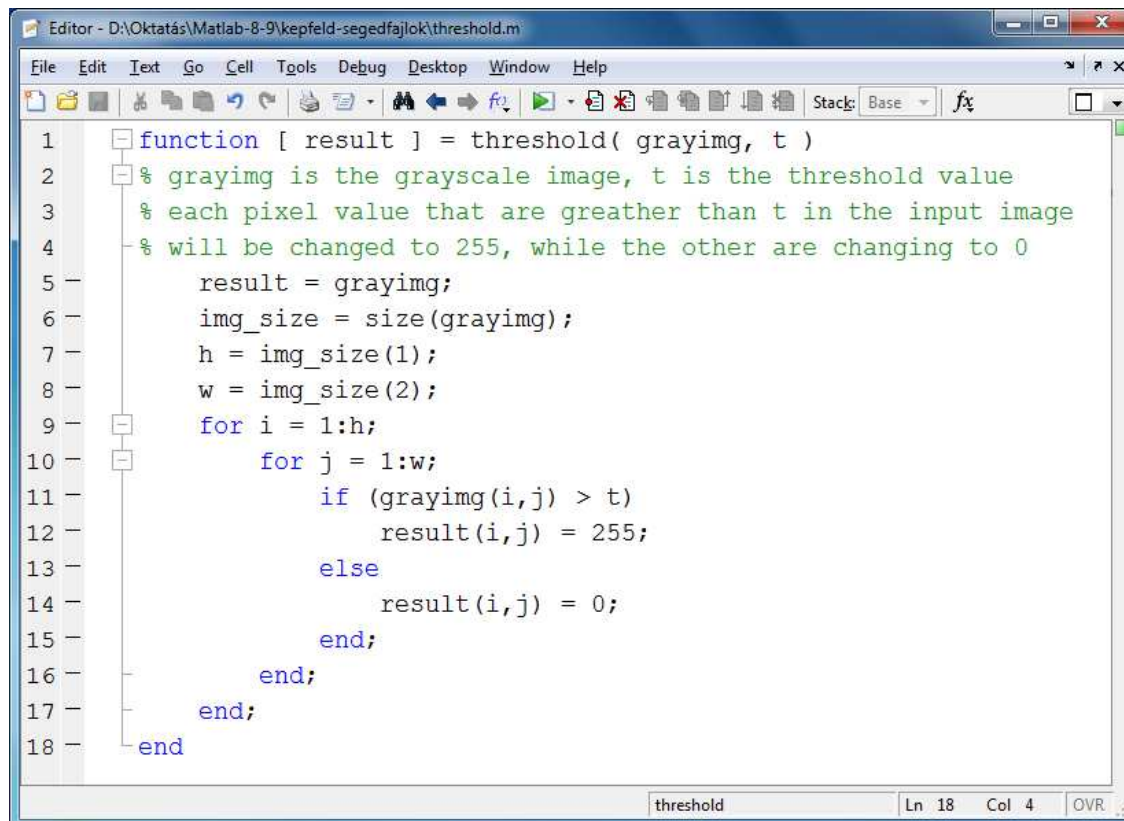
- Ciklusok egymásba is ágyazhatók
- Egymásba ágyazott for ciklusok tipikus alkalmazása: mátrix adatokkal való feltöltése, elemeinek bejárása, módosítása
- Példa: Hilbert-mátrix előállítása
 - (Matlabban persze *nem* ez a hatékony mo.)
 - Eml.: Tudjuk, hogy létezik beépített parancs
- A kód (4×4-es mátrix):
H4P = [];
for i = 1:4
 for j = 1:4
 H4P(i,j) = 1/(i+j-1);
 end
end
H4P
- Megj.: A mátrixot manuálisan for ciklusok nélkül is elő tudjuk állítani
> H4 = [1./[1:4];1./[2:5];1./[3:6];1./[4:7]]

```
Command Window
File Edit Debug Desktop Window Help
>> H4P=[];
    for i=1:4
        for j=1:4
            H4P(i,j)=1/(i+j-1);
        end
    end
H4P
H4P =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
fx >> |
```

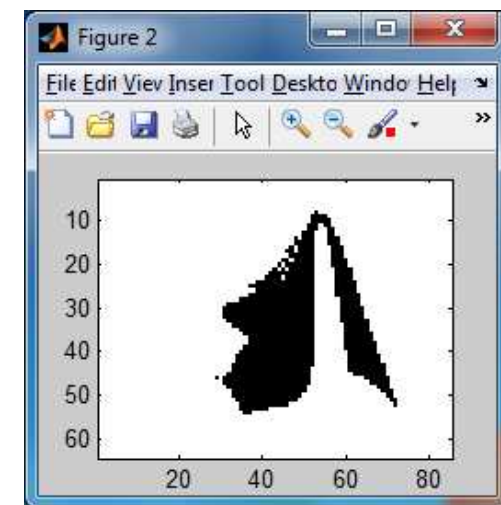
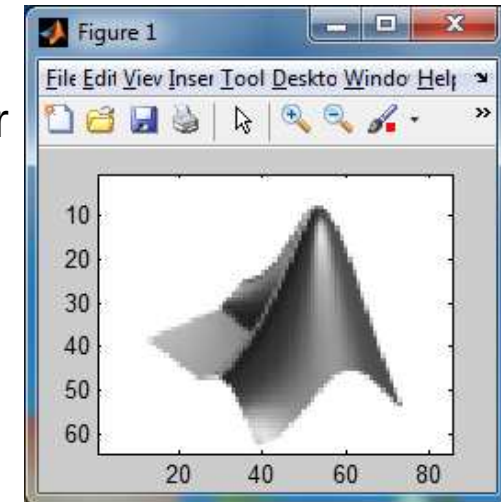


Ciklusok

- Egymásba ágyazott ciklusok alkalmazása (folyt.)
- Eml.: Vágási feladat (szürkeárnyaltos képre)
 - Ha a színérték bizonyos korlát felett van, akkor fehér lesz, különben fekete (kontraszt szélsőséges kiemelése)
 - Lásd még: vektorizáció



```
1 function [ result ] = threshold( grayimg, t )
2 % grayimg is the grayscale image, t is the threshold value
3 % each pixel value that are greather than t in the input image
4 % will be changed to 255, while the other are changing to 0
5 result = grayimg;
6 img_size = size(grayimg);
7 h = img_size(1);
8 w = img_size(2);
9 for i = 1:h;
10     for j = 1:w;
11         if (grayimg(i,j) > t)
12             result(i,j) = 255;
13         else
14             result(i,j) = 0;
15         end;
16     end;
17 end;
18 end
```





Ciklusok

While ciklus

- Motiváció: Nem tudjuk/akarjuk előre megmondani, hogy pontosan hányszor kell lefuttatni a ciklusmagot
 - Ez függhet például az inputtól
- Ilyenkor bennmaradási feltétel fogalmazható meg (igaz feltétel)
 - A ciklusváltozót (ha van) nekünk kell módosítani
- Lehetséges hiba: végtelen ciklust írunk
 - Ctrl+C-vel tudjuk leállítani (jó esetben)
- Szintaktika:
`while` logikai kif. [, | ;] utasítás(ok), `end`
 - Persze a cikluson kívül (előtte és utána) megfelelő tevékenységek adandók meg
- Példák
 - Gépi epszilon meghatározása
> `eps = 1; while (1+eps) > 1, eps = eps/2; end, eps = eps*2`
 - Tudjuk: `eps` már eredetileg beépített változóként is létezik (és itt felülírjuk)
 - Melyik Hilbert mátrix determinánsa lesz először kisebb `eps`-nél?
> `i=1; while det(hilb(i))>eps; i=i+1; end, i`



Ciklusok

- While ciklus példa: szinusz értékek előállítása Taylor-sorral (script)

- Képlet:
$$\sin x = \frac{x}{1!} - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

- Lépések

- A következő tag előjelváltó, $x \cdot x$ -szel szorzódik, $n \cdot (n-1)$ -gyel osztódik
- Ha a kívánt pontosságot (eps) elértük, kiszállunk

The screenshot shows the MATLAB Editor window with a script named 'sinusz.m' and the Command Window displaying the results of running the script.

Editor - D:\Oktatás\Matlab-8-9\sinusz.m

```
1 % Változók törlése, long kijelzés
2 clear all;
3 format long;
4 fok=[30 45 60];
5 x=fok*pi/180;
6 % Csupa 0 mátrix az eredménynek
7 s=zeros(1, length(x)); t=x; n=3;
8 % Ciklus
9 while max(abs(t))>eps
10     s=s+t; t=-t.*x.^2/(n*(n-1)); n=n+2;
11 end;
12 % Eredmények kiírása
13 fok, s, sinus = sin(x)
```

Command Window

```
>> sinusz
fok =
    30    45    60
s =
    0.500000000000000    0.707106781186547    0.866025403784438
sinus =
    0.500000000000000    0.707106781186547    0.866025403784439
fx >>
```

script Ln 14 Col 1 OVR



Ciklusok

Specialitások

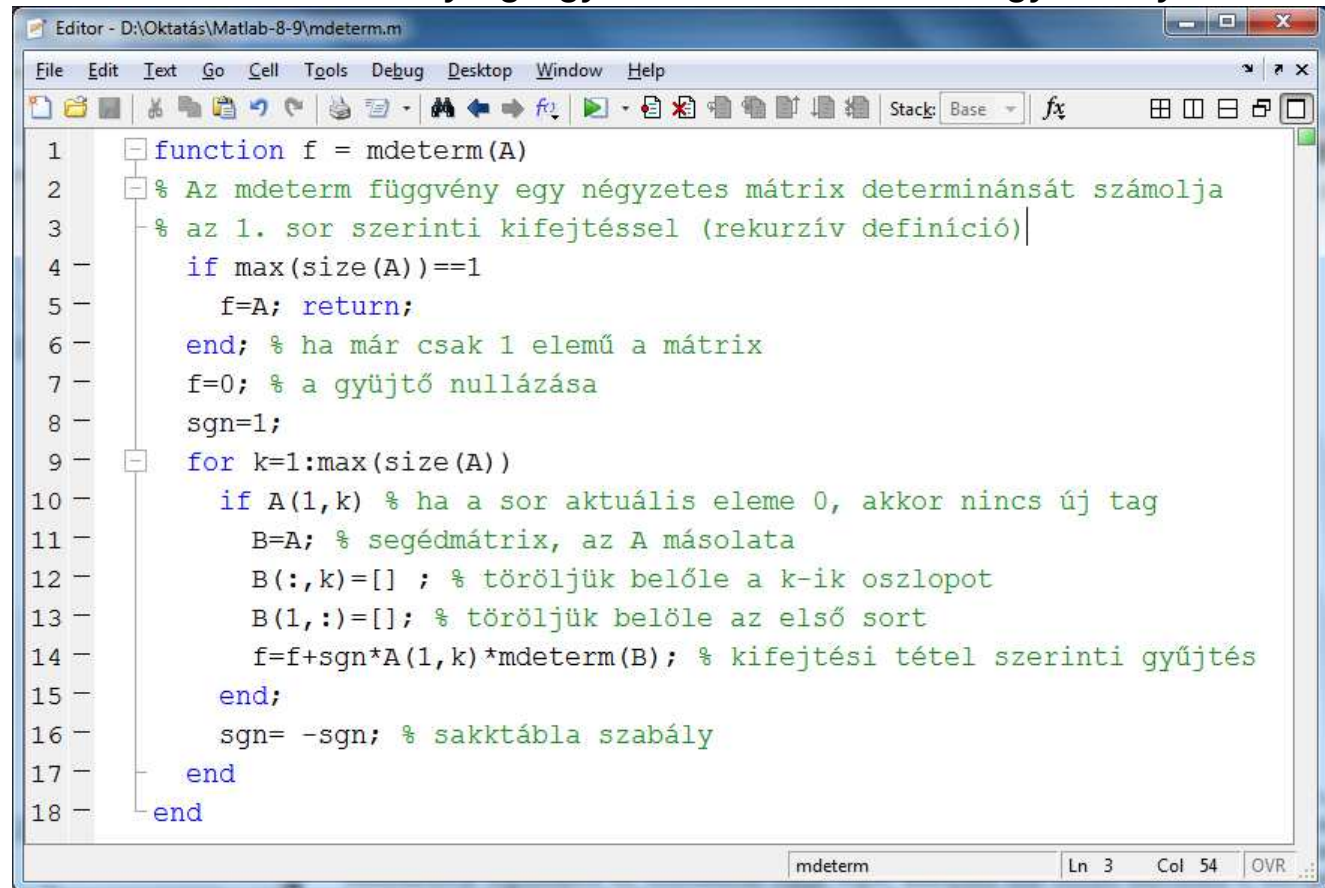
- For ciklusnál lépésköz megadható (tudjuk)
- A ciklusváltozó tetszőleges sorvektoron is végigmehet (és elemeinek értékeit felveheti)
 - Így akár oszlopvektorok is kerülhetnek a ciklusváltozóba
- Példa 1.
> `for i=[pi, inf, nan] i, end`
 - Mi történne akkor, ha pontosvesszőt raknánk az elemek közé?
(Mi lenne i értéke?)
- Példa 2.
> `A = rand(2,4), for x=A x, end;`
 - Mi kerül most az x változóba?
- A Matlabban ciklusok párhuzamos végrehajtására is van lehetőség (*parfor* szerkezet), de ehhez kell a Parallel Computing Toolbox
 - Lásd: Sógó
- Nem strukturált vezérlő eszközök is léteznek a Matlabban
 - Break, continue (nem tanuljuk)



Rekurzió

- A függvény ilyenkor saját magát hívja
 - A megfelelő változók (adatok) veremben tárolódnak, ennek mélysége korlátozott (módosítható)
- Klasszikus, rekurzívan megoldható feladatok: faktoriális számítás, determináns kifejtés, Hanoi tornyai
 - A rekurzív szerkezet sok esetben viszonylag egyszerűen kiváltható hagyományos iterációval (lásd elektr. jegyzet)

- Példa:
Determináns meghatározása rekurzív módon (saját fv.)
- Elemezzük az eljárást!
- Ellenőrizzük a helyes működést, és hasonlítsuk össze a beépített *det* függvénnyel



```
Editor - D:\Oktatás\Matlab-8-9\mdeterm.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function f = mdeterm(A)
2 % Az mdeterm függvény egy négyzetes mátrix determinánsát számolja
3 % az 1. sor szerinti kifejtéssel (rekurzív definíció)
4 if max(size(A)) == 1
5     f=A; return;
6 end; % ha már csak 1 elemű a mátrix
7 f=0; % a gyűjtő nullázása
8 sgn=1;
9 for k=1:max(size(A))
10     if A(1,k) % ha a sor aktuális eleme 0, akkor nincs új tag
11         B=A; % segédmátrix, az A másolata
12         B(:,k)=[]; % töröljük belőle a k-ik oszlopot
13         B(1,:)=[]; % töröljük belőle az első sort
14         f=f+sgn*A(1,k)*mdeterm(B); % kifejtési tétel szerinti gyűjtés
15     end;
16     sgn= -sgn; % sakktábla szabály
17 end
18 end
```

mdeterm Ln 3 Col 54 OVR





Adatbekérés és kiírás

Bekérés

- Szintaktika: `változó = input()`
 - A parancs az argumentumként megadott üzenet kiadása után a felhasználói begépelést eltárolja a változóban
 - Példa (*switch* szerkezetnél):
> `n = uint16(input('Kérek egy pozitív egészet (max. 65535): '));`
- Grafikus input: `ginput`
 - Egérkoordinátákat és gombot (egér vagy billentyű) fogad és tárol

Kiírás

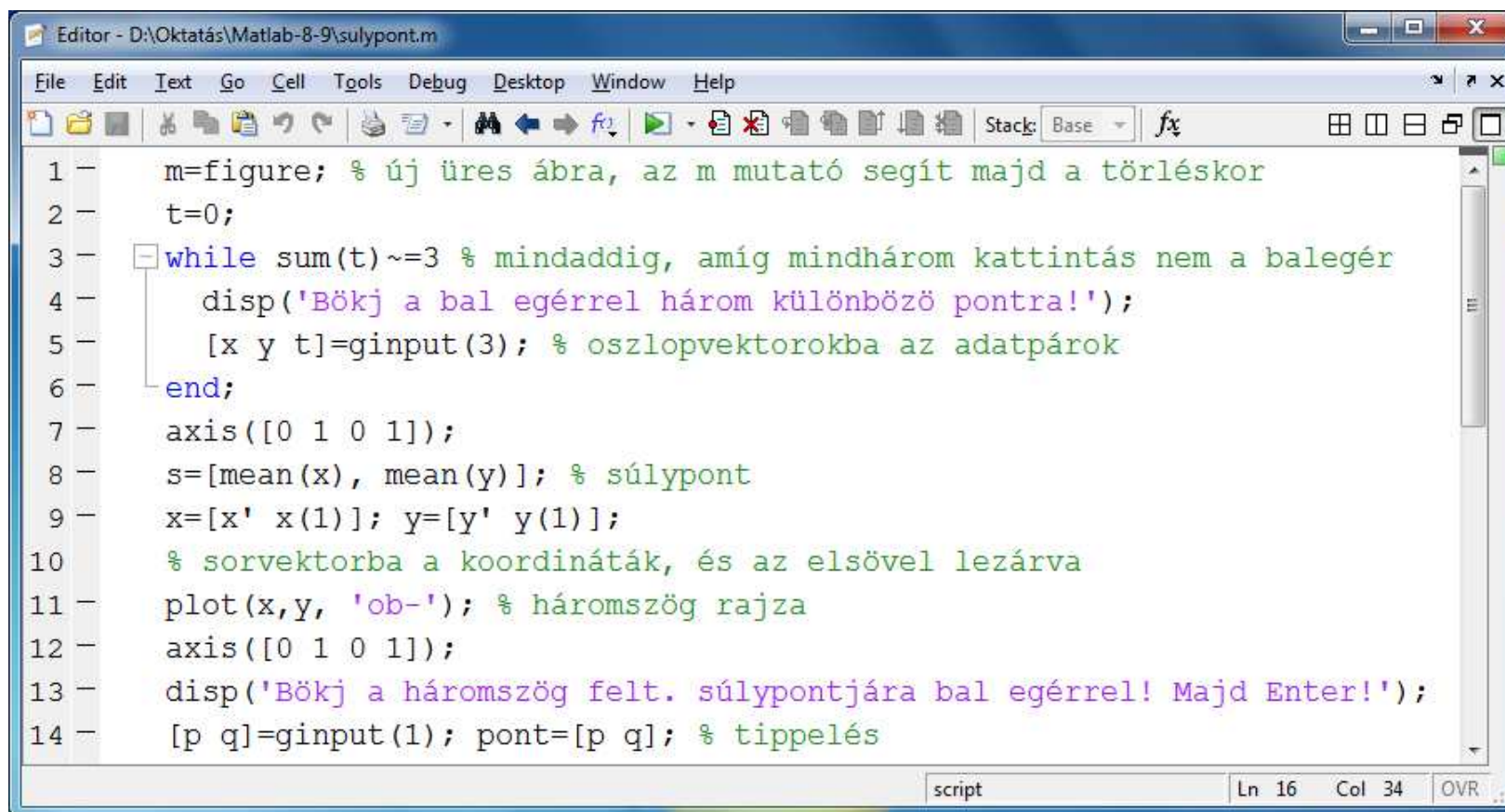
- A *disp* parancs egy változó értékét írja ki a változó neve nélkül
 - Konverzió szükséges lehet (pl. *num2str*)
 - Példa (*switch* szerkezetnél):
> `disp(strcat(num2str(n), ' osztható 4-gyel'));`
- C-s jellegűek az *fprintf* és *sprintf* függvények
 - *fprintf* – szöveges adatfolyamot állít elő, ami ha nem fájlba irányul, akkor a Command Window-ba kerül
 - *sprintf* – szintén szöveget állít elő, de az az *ans* változóba kerül
- Grafikus ábrán a *text* függvénnyel lehet szöveget elhelyezni (tudjuk)



Interaktív grafika

Feladat (játék):

- Egy üres ábra felületén megadunk az egérrel három pontot. A felrajzolt háromszög súlypontjára kell a lehető legjobban tippelve kattintani. A súlypont megcélzása után felrajzoljuk a tippelés helyét, a tényleges súlypontot és minősítést is adunk.



```
1 - m=figure; % új üres ábra, az m mutató segít majd a törléskor
2 - t=0;
3 - while sum(t)~=3 % mindaddig, amíg mindhárom kattintás nem a balegér
4 -     disp('Bökj a bal egérrel három különböző pontra!');
5 -     [x y t]=ginput(3); % oszlopvektorokba az adatpárok
6 - end;
7 - axis([0 1 0 1]);
8 - s=[mean(x), mean(y)]; % súlypont
9 - x=[x' x(1)]; y=[y' y(1)];
10 - % sorvektorba a koordináták, és az elsővel lezárva
11 - plot(x,y, 'ob-'); % háromszög rajza
12 - axis([0 1 0 1]);
13 - disp('Bökj a háromszög felt. súlypontjára bal egérrel! Majd Enter!');
14 - [p q]=ginput(1); pont=[p q]; % tippelés
```



Interaktív grafika

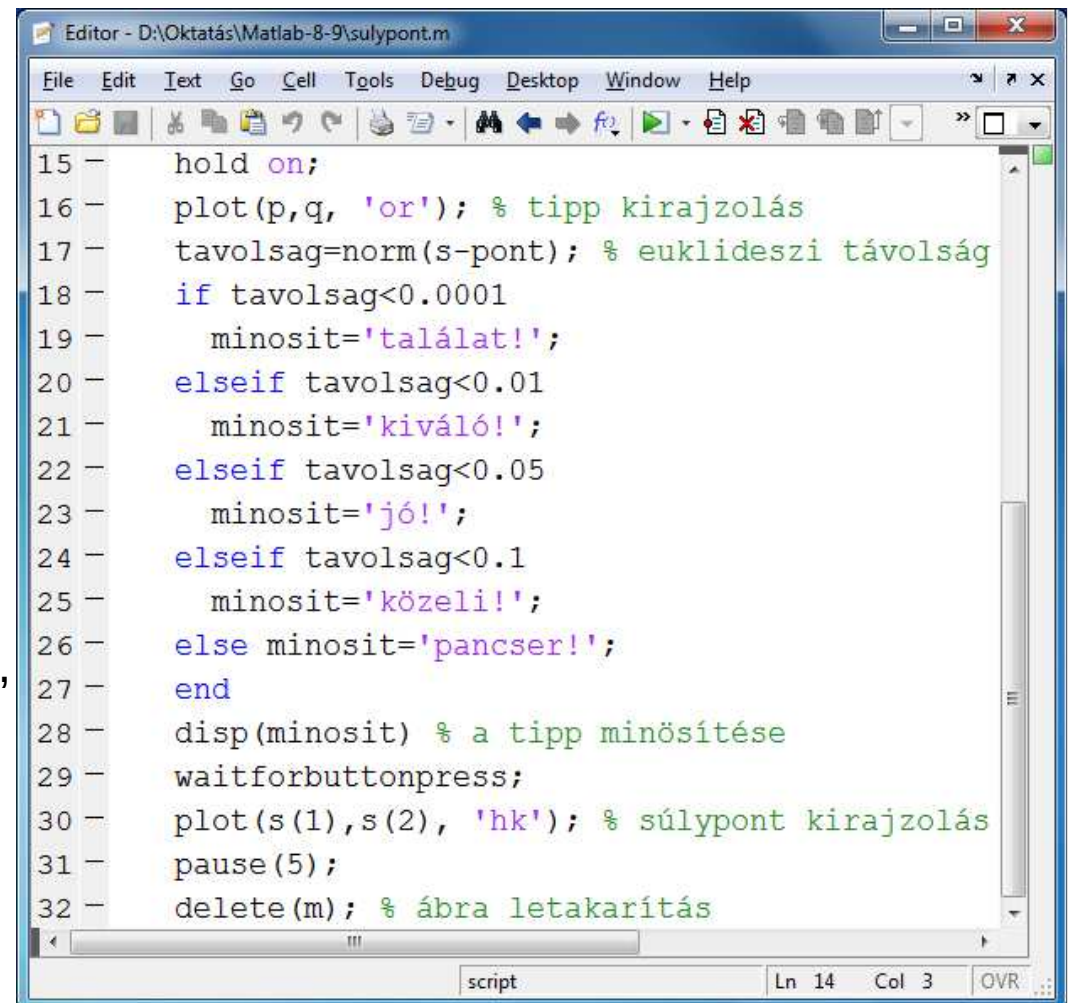
Játék (folyt.), elemzés

- *ginput* szintaktika:

`[x_koordináták, y_koordináták, billentyűkódok] = ginput(n)`

- Ahol *n* a kért kattintások száma, az output elemek pedig oszlopvektorok. A beolvasott koordináták a grafika kirajzolt intervallumába esnek.
- A kapott billentyűkódok:
 - 1 – bal ...
 - 2 – középső ...
 - 3 – jobb egérgombot nyomtunk
 - egyéb – a lenyomott billentyű ASC kódja

- Figyeljük meg a ciklusszervezést, a szelekciót, a kiírásokat és a grafikai és egyéb elemek használatát!



```
Editor - D:\Oktatás\Matlab-8-9\sulypont.m
File Edit Text Go Cell Tools Debug Desktop Window Help
15 - hold on;
16 - plot(p,q, 'or'); % tipp kirajzolás
17 - tavolsag=norm(s-pont); % euklideszi távolság
18 - if tavolsag<0.0001
19 -     minosit='találat!';
20 - elseif tavolsag<0.01
21 -     minosit='kiváló!';
22 - elseif tavolsag<0.05
23 -     minosit='jó!';
24 - elseif tavolsag<0.1
25 -     minosit='közele!';
26 - else minosit='pancser!';
27 - end
28 - disp(minosit) % a tipp minősítése
29 - waitforbuttonpress;
30 - plot(s(1),s(2), 'hk'); % súlypont kirajzolás
31 - pause(5);
32 - delete(m); % ábra letakarítás
script Ln 14 Col 3 OVR
```



Paraméterek kezelése

- A Matlab függvényekben a paraméterek (aktuális) száma és esetleg típusa is bizonyos keretek között változhat
 - Például nem kötelező mindig minden aktuális paramétert megadni
- Fontosabb paraméterkezelő függvények
 - *nargin*, *nargout* – input/output argumentumok száma
- *error* – hibakezelés

The screenshot shows the MATLAB Editor window with a file named `add.m`. The code defines a function `add` that takes three arguments `x`, `y`, and `z`. It uses `nargin` to check the number of input arguments. If fewer than 2 arguments are provided, it throws an error. If exactly 2 arguments are provided, it adds them. If 3 arguments are provided, it adds all three.

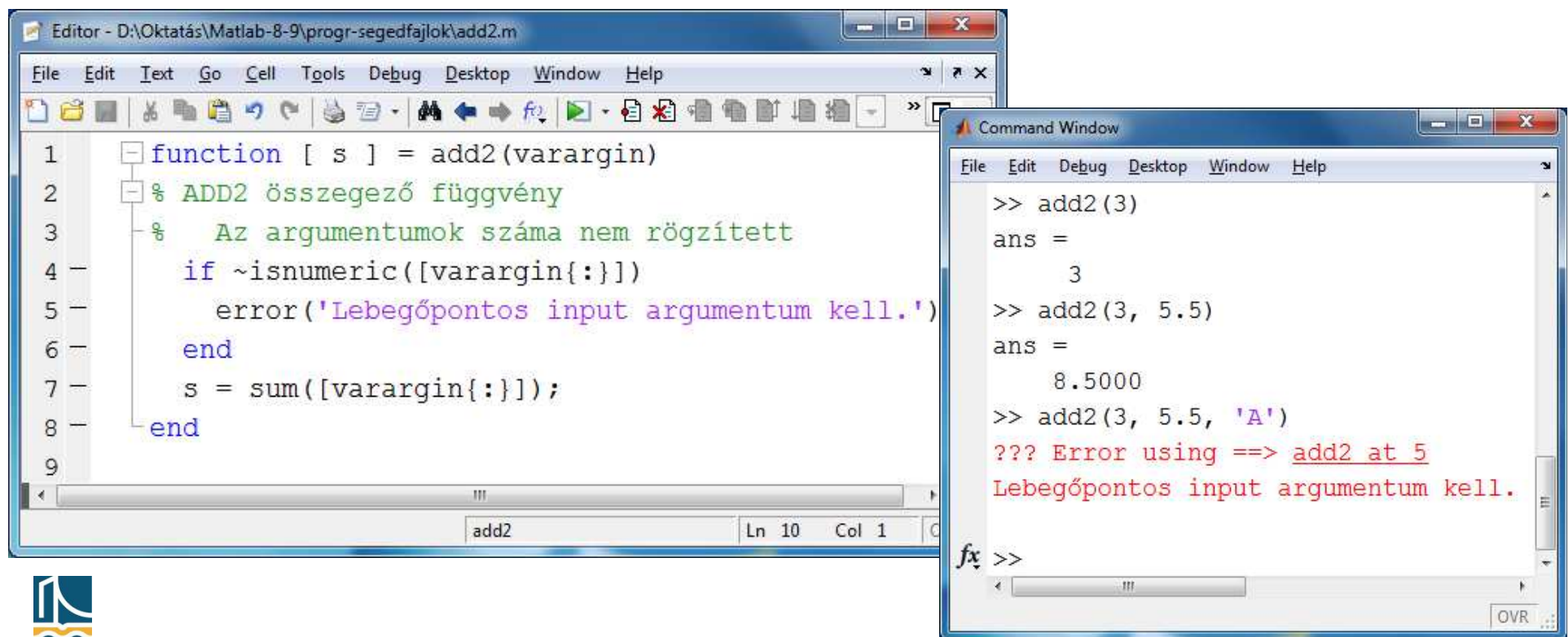
```
1 function [ s ] = add( x, y, z )
2 % ADD összegező függvény
3 % 2 vagy 3 argumentum adható meg
4 if nargin < 2
5     error('Legalább két input argumentum kell.')
6 end
7 if nargin == 2
8     s = x + y;
9 else
10    s = x + y + z;
11 end
12 end
```

The Command Window shows the execution of the function. The first call `>> add(2, 3)` returns `ans = 5`. The second call `>> add(2)` results in an error message: `??? Error using ==> add at 5` and `Legalább két input argumentum kell.`

```
>> add(2, 3)
ans =
     5
>> add(2)
??? Error using ==> add at 5
Legalább két input argumentum kell.
fx >>
```

Paraméterek kezelése

- Fontosabb paraméterkezelő függvények (folyt.)
 - *varargin* – változó számú input paraméterek listáját adja (hasonló: *varargout*); az eredmény cellatömb
 - A *varargin{:}* kifejezés vesszővel elválasztott listát ad vissza, konverzió kell
 - Az argumentumok típusát/értékét sok esetben ellenőrizni kell; erre szolgál többek között: *isnumeric*, *ischar* (*isstr*), *isinf*, *isnan* stb.



The screenshot displays the MATLAB environment. The Editor window shows a function named `add2` that uses `varargin` to handle a variable number of input arguments. The function checks if all inputs are numeric using `isnumeric` and throws an error if not. The Command Window shows the function being called with different arguments: a single number, two numbers, and a number followed by a string, demonstrating the error handling.

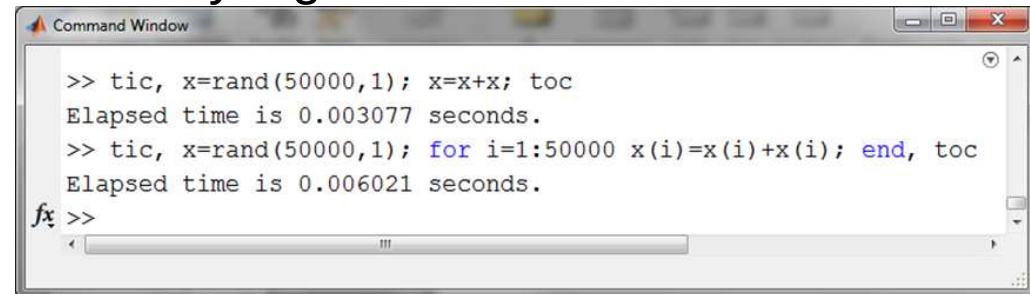
```
Editor - D:\Oktatás\Matlab-8-9\progr-segedfajlok\add2.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 function [ s ] = add2(varargin)
2 % ADD2 összegező függvény
3 % Az argumentumok száma nem rögzített
4 if ~isnumeric([varargin{:}])
5     error('Lebegőpontos input argumentum kell.')
6 end
7 s = sum([varargin{:}]);
8 end
9

Command Window
File Edit Debug Desktop Window Help
>> add2(3)
ans =
     3
>> add2(3, 5.5)
ans =
    8.5000
>> add2(3, 5.5, 'A')
??? Error using ==> add2 at 5
Lebegőpontos input argumentum kell.
fx >>
```

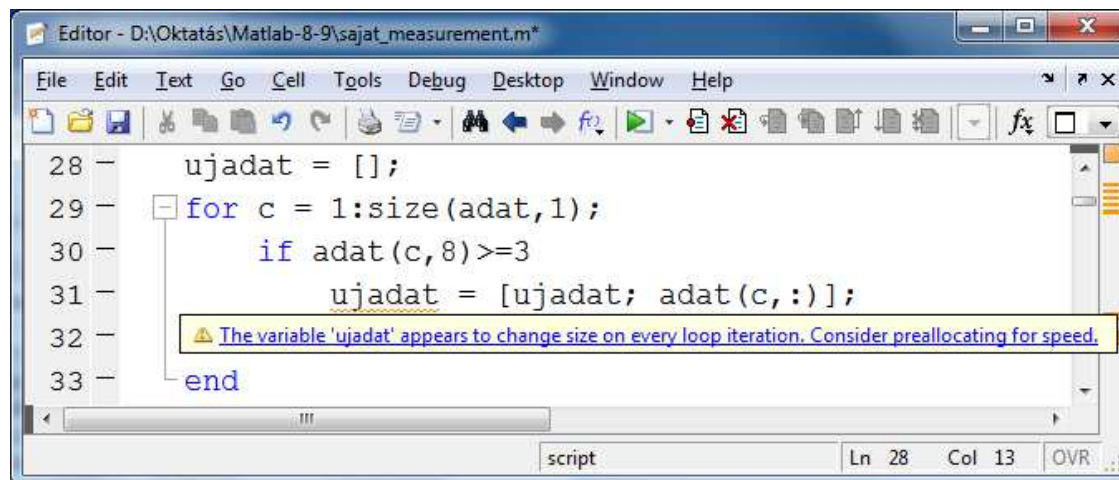
A Matlab kód hatékonysága

Néhány általános jó tanács a kód hatékonyságának növeléséhez

- *for* ciklusok vizsgálata, „vektorizáció”
 - A ciklus nem hatékonyan olvas/ír
 - A Matlab ellenben nagyon hatékony a mátrixokkal és a vektorokkal végzett műveletek végrehajtásában
- Memória előfoglalás
- Warning üzenetek elemzése és kiküszöbölése

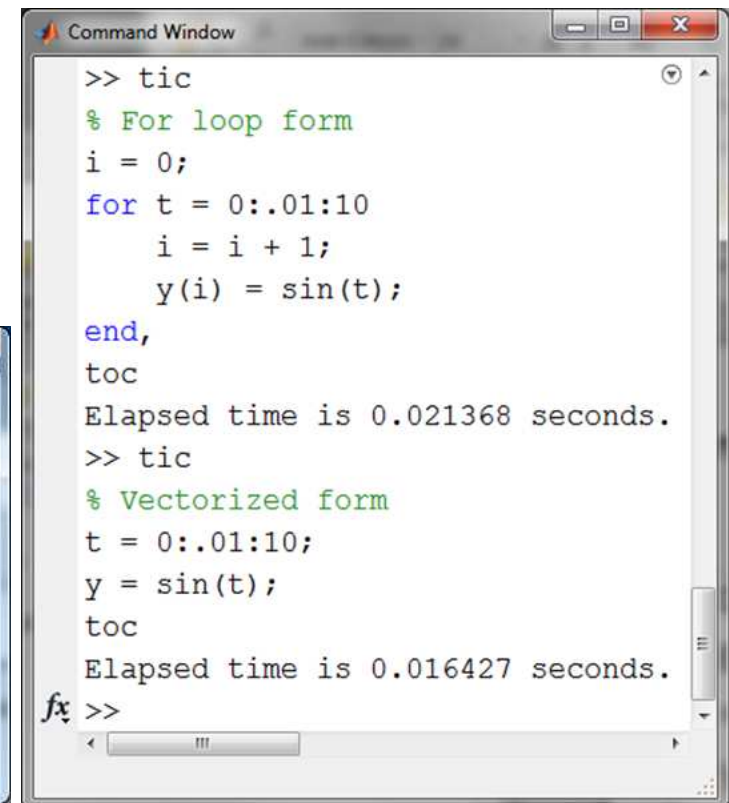


```
>> tic, x=rand(50000,1); x=x+x; toc
Elapsed time is 0.003077 seconds.
>> tic, x=rand(50000,1); for i=1:50000 x(i)=x(i)+x(i); end, toc
Elapsed time is 0.006021 seconds.
fx >>
```



```
28 - ujadat = [];
29 - for c = 1:size(adat,1);
30 -     if adat(c,8)>=3
31 -         ujadat = [ujadat; adat(c,:)];
32 -
33 -     end
```

The variable 'ujadat' appears to change size on every loop iteration. Consider preallocating for speed.



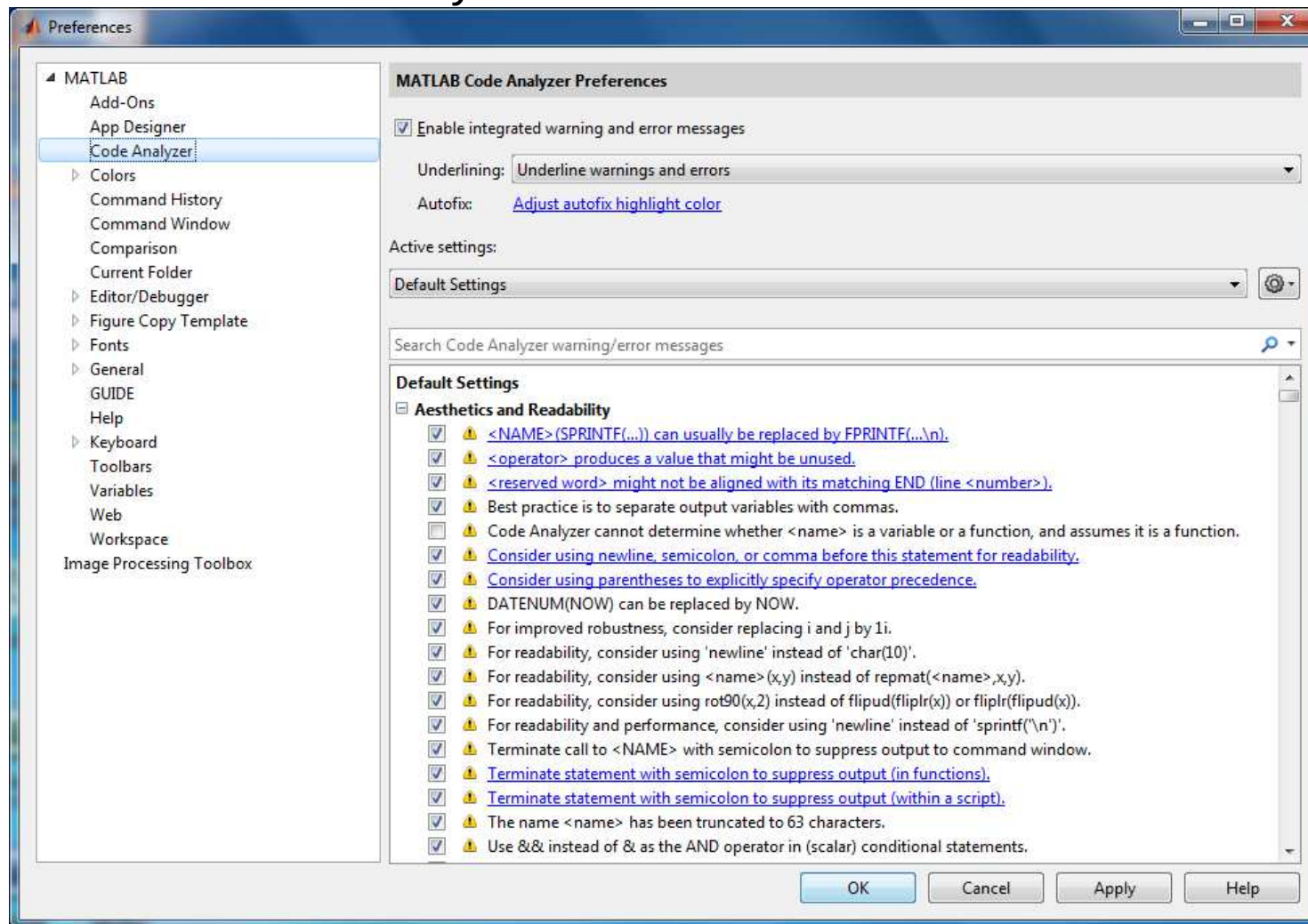
```
>> tic
% For loop form
i = 0;
for t = 0:.01:10
    i = i + 1;
    y(i) = sin(t);
end,
toc
Elapsed time is 0.021368 seconds.
>> tic
% Vectorized form
t = 0:.01:10;
y = sin(t);
toc
Elapsed time is 0.016427 seconds.
fx >>
```



A Matlab kód hatékonysága

Fordító (kódelemző) üzenetek

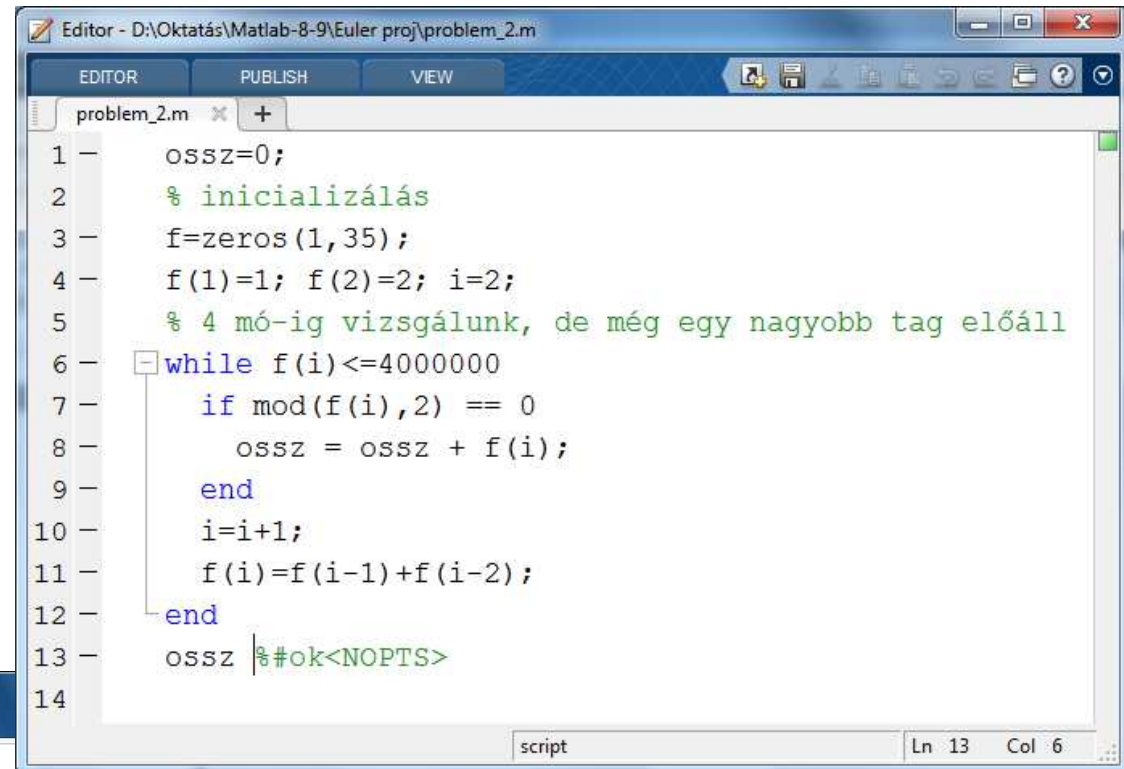
- Érdekes őket tanulmányozni!



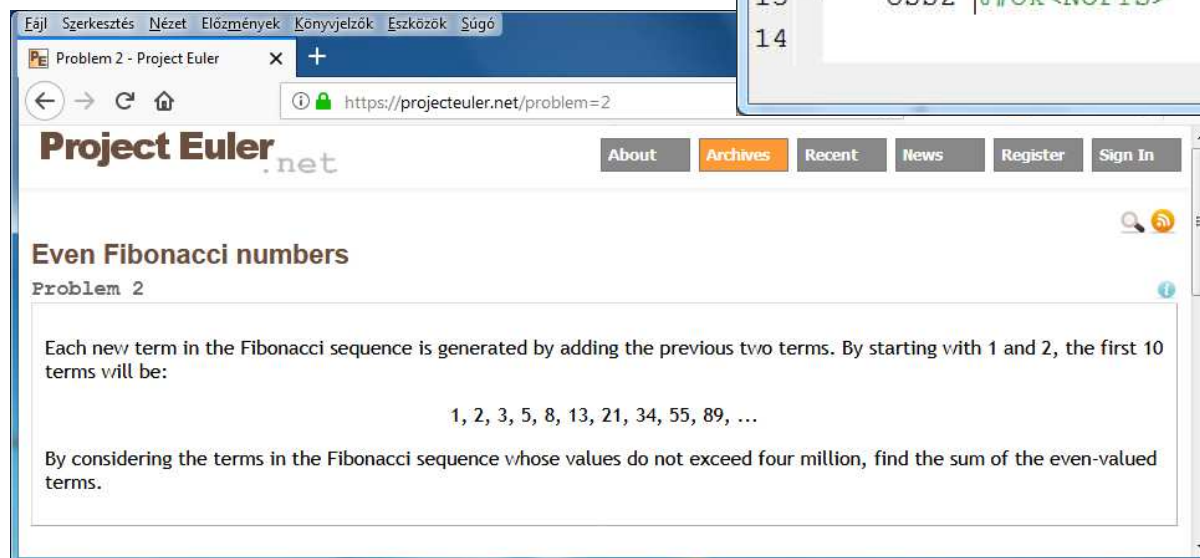
Haladó programozás, „verseny”

Euler-projekt

- Fokozatosan (viszonylag gyorsan) nehezedő feladatok
- Egy idő múlva komoly kihívás!



```
Editor - D:\Oktatás\Matlab-8-9\Euler proj\problem_2.m
EDITOR PUBLISH VIEW
problem_2.m x +
1 - ossz=0;
2 - % inicializálás
3 - f=zeros(1,35);
4 - f(1)=1; f(2)=2; i=2;
5 - % 4 millió-ig vizsgálunk, de még egy nagyobb tag előáll
6 - while f(i)<=4000000
7 -     if mod(f(i),2) == 0
8 -         ossz = ossz + f(i);
9 -     end
10 -     i=i+1;
11 -     f(i)=f(i-1)+f(i-2);
12 - end
13 - ossz % #ok<NOPTS>
14
```



Project Euler .net

About Archives Recent News Register Sign In

Even Fibonacci numbers

Problem 2

Each new term in the Fibonacci sequence is generated by adding the previous two terms. By starting with 1 and 2, the first 10 terms will be:

1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

By considering the terms in the Fibonacci sequence whose values do not exceed four million, find the sum of the even-valued terms.