```
%
aa = [1 2 3];
%
aa = [1 2 3]   % brak „;” na koncu wiersza, przez co polecenie
               % wyswietli rowniez nowa tresc zmiennej.
               % proszĘ unikaĆ uŻywania polskich znakÓw w kodzie
aa = [1 ...    % kontynuowanie polecenia w kolejnym wierszu
 2 3]
%
aa = [10:-2:0]
```

Zadanie. Stwórz wektor o wartościach 1, 0.9, 0.8 … -0.9

Obracanie wektora lub macierzy

```
aa = [1:3 ; 11:13]; % znak „ ; ” mowi o zlaczeniu elementow
                    % „w pionie”
                    % (albo po prostu przejsciu
                    % do nastepnego wiersza).
aa'                 % transponowanie
```

Zadanie. Stwórz macierz o poniższej zawartości, pisząc 1 linię kodu:

1 2.1
4 2.0
..
40 0.8

Operacje „z kropką” i bez na macierzach

```
aa .* 10            % tzw. operatory z kropka i skalarem,
                    % czyli na kazdym elemencie
                    % bedzie dokonywana operacja oddzielnie
aa ./ [1 2 10]
aa +  [10 ; 100]    % wektor po prawej stronie jest transponowany,
                    % operacje beda wykonywane dla kazdego
                    % wiersza osobno,
                    % pierwszy wiersz będzie powiększony o 10,
                    % drugi o 100
aa +  [10 100]      % wektor po prawej stronie
                    % nie jest transponowany,
                    % operacje beda wykonywane dla kazdej
                    % kolumny osobno,
                    % pierwsza kolumna będzie powiększona o 10,
                    % druga o 100
aa .^ 0.5           % wszystkie elementy podnies do potęgi 0.5,
                    % czyli spierwiastkuj
```

Zadanie. W poniższej macierzy pomnóż **wiersze** kolejno przez 1, 100, 10 za pomocą jednego polecenia.

Macierz wejściowa: Wynik:

1	2	3	1	2	3
4	5	6	400	500	600
7	8	9	70	80	90

Zadanie. W poniższej macierzy pomnóż **kolumny** kolejno przez 1, 0.1, 10 za pomocą jednego polecenia.

Macierz wejściowa:			Wynik:		
1	2	3	1	0.2	30
4	5	6	4	0.5	60
7	8	9	7	0.8	90

Logiczne operacje na macierzach

```
aa > 1 & aa ~= 11   % zwroci wektor/macierz
                    % z wartosciami bool'owskimi (logicznymi)
```

Zadanie. Stwórz wektor o wartościach 11, 12 .. 20 i wskaż czy kolejne elementy w tym wektorze są jednocześnie większe niż 15 i mniejsze lub równe 18. Wynik powinien wynosić 0, 0, 0, 0, 0, 1, 1, 1, 0, 0.

Mnożenie

```
aa * [1 10 100]'   % podczas mnozenie wektorow lub macierzy
                   % za pomocą operatora „*" bez kropki następuje
                   % zwyczajne mnozenie macierzowe
                   % (tzw. „wiersze przez kolumny").
```

Zadanie. Sprawdź wynik mnożenia poniżej macierzy przez poniższy wektor transponowany („pionowy")

Macierz		wektor transponowany
1	2.1	1
3	2.2	1000
..		
41	4.1	

Odwracanie macierzy

```
[1 2;3 4]^-1
```

Zadanie. Za pomocą odwracania macierzy wylicz wartość układu równań $3 \cdot x_1 + 1 \cdot x_2 = 31; -x_1 + x_2 = -9$, czyli $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ -1 & 1 \end{bmatrix}^{-1} * \begin{bmatrix} 31 \\ -9 \end{bmatrix}$, powinno wyjść $x_1 = 10; x_2 = 1$.

Indeksowanie elementów macierzy 2d

```
% pierwszy element ma numer 1, a ostatni „end”
aa(:, 2:end-1)      % wszystkie wiersze ( : ),
                    % kolumny od drugiej do przedostatniej
```

Zadanie. Dla poniższej macierzy pokaż wartości wszystkich kolumn zaczynając, a wiersza zaczynając od przedostatniego i kończąc na pierwszym.

Macierz wejściowa:			Wynik:		
1	2	3	4	5	6
4	5	6	1	2	3
7	8	9			

Indeksowanie macierzy pojedynczym wektorem

```
aa(1:4)   % w przypadku macierzy (nie wektora)
          % i indeksowania wektorem
          % elementy sa liczone kolumnami,
          % odradza się jawne indeksowanie w ten sposób.
```

Różne sposoby indeksowanie wektora

```
wek = [11:14];
wek(wek > 12)             % zwroci elementy > 12
wybor_bool = wek > 12     % zwroci informacje, ktore elementy
                          % spelniaja warunek,
                          % czyli [0 0 1 1]
wek(wybor_bool)           % indeksowanie przy pomocy wartosci bool,
                          % zwroci elementy > 12
wybor_idx = find(wek > 12)    % zwroci indeksy elementow,
                              % ktore spelniaja warunek,
                              % czyli [3 4]
wek(wybor_idx)            % indeksowanie przy pomocy indeksow,
                          % zwroci elementy > 12
```

Inne indeksowanie macierzy

```
aa = [1 2 3;11 12 13]
aa > 2          % zwroci informacje ktore el. spelniaja warunek,
                % czyli tutaj to [0 0 1; 1 1 1]
aa(aa > 2)      % elementy > 2, zwroci wektor transponowany
                %  [11; 12; 3; 13]

find(aa > 2)    % indeksy elementow > 2 w postaci wektora transp.,
                % czyli [2; 4; 5; 6], odradzany sposob indeksowania

bb=aa; bb(aa<2) = -9; bb
```

Zadanie. W poniższej macierzy powiększ wartości pomiędzy 2 i 5 o 10 z pomocą jednej linii kodu.

Macierz wejściowa: Wynik:

1	2	3	1	2	13
4	5	6	14	5	6
7	8	9	7	8	9

Wybrane operacje "agregacji" na macierzach 2D

```
aa = [1:3; 11:13]
min(aa)                  % wyszukanie minimalnej wartosci
                         % dla kazdej kolumny
[w,i] = min([33 11 22])  % w= minimalna wartosc=11;
                         % i= indeks tej wartosci=2
[~, i] = min([33 11 22]) % pobierz tylko DRUGI parametr
                         % zwracany przez funkcje min,
                         % czyli index minimalnego elementu
max(aa)        %
mean(aa, 2)    % wyszukanie sredniej w kazdym wierszu
               % (tutaj „2” okresla numer agregowanego wymiaru)
sum(aa, 1)     % wyliczenie sumy elementow w kazdej kolumnie
sum(aa)
```

Zadanie. Proszę stworzyć macierz widoczną poniżej, następnie w każdej kolumnie osobno pomniejszyć wartości o średnią wartość w tej kolumnie.

Macierz wejściowa:			Wynik:		
1	2	3	-3	-3	-1
4	5	4	0	0	0
7	8	5	3	3	1

```
size(aa, 1)              % rozmiar "pierwszego wymiaru" macierzy
length([10:0.1:30])      % dlugosc wektora,
length([1:3; 11:13])     % szerokosc macierzy
```

Inne operacje

```
doc clear
clear bb;      % kasowanie zmiennej (dosc wolna operacja)
clear ;        % kasowanie wszystkich zmiennych
rand(2, 3);         % macierz 2x3 z wartosciami losowymi z
przedzialu (0; 1)
sort(rand(1,10) );
randperm(5);        % losowa permutacja liczb 1:n,
                    % na przykład:  2, 3, 5, 1, 4
```

Zadanie. Stwórz ręcznie macierz widoczną poniżej, następnie pobierz wielkość macierzy poniżej, później stwórz macierz o tej samej wartości wypełnioną wartościami losowymi z przedziału [-1;+1].

Macierz wejściowa:			Przykładowy wynik:		
1	2	3	-0.393	0.421	0.831

4 5 4 0.315 -0.790 -0.013

Pętla **while**

```
aa = [1:3;11:13]; bb = 1; cc = 0;
while (bb <= size(aa, 2))
  tmp = cc + aa(1, bb);
  cc = tmp;
  bb = bb + 1;
end;
tmp      % zmienna nie jest usuwana po wyjsciu z petli
cc
bb
clear bb cc;
```

Pętla **for**

```
aa = [1:3;11:13]; suma = 0;
for i = 1:size(aa, 2)     % najpopularniejsza skladnia to
                          % zmienna = wektor nietransponowany
  suma = suma + aa(1, i);
     % petla uruchamiana dla zmiennej „i”
     % rownej kolejnym wartosciom wektora 1:size(..) ...
     % Zmienna „i” bedzie w kolejnych iteracjach zawierala
     % kolejne wartosci, czyli tutaj 1,2,3.
     % Ta petla wykona się length(1:size(aa, 2)) razy.
end;
suma
clear suma i       % usuniecie z pamieci tych zmiennych
```

Uwaga! Pętla for, do zmiennej wyznaczonej (tutaj *i*) będzie w kolejnych iteracjach podstawiać kolejne kolumny a nie wartości macierzy. Szczegóły widoczne są w przykładach poniżej.

```
mac = [1 2 3; 4 5 6]; wek_pion = [11 12 13]';
wek_poz = [11 12 13];
for i = mac
  i       % petla wykona się 3 razy,
          % zmienna i bedzie zawierala w kolejnych iteracjach
          % transponowane wektory,
          % czyli [1; 4], [2; 5], [3; 6]
end;

for i = wek_pion
  i       % petla wykona się 1 raz,
          % zmienna i bedzie zawierala pionowy wektor [11; 12; 13]
end;

for i = wek_poz
  i       % petla wykona się 3 razy,
          % zmienna i bedzie zawierala w kolejnych iteracjach
wartosci 11, 12, 13
```

```
end;
```

Zadanie. Proszę stworzyć macierz o rozmiarze 4x3 wypełnioną losowymi wartościami całkowitymi z przedziału [0; 5], a następnie proszę za pomocą podwójnej pętli przeiterować każdy wiersz osobno od strony lewej po prawą i sprawić, by kolumny od drugiej do ostatniej były powiększone o wartość po ich lewej stronie (suma z kumulacją). Przydatne funkcje to **rand, round, ceil, floor.** Na przykład:

Macierz 4x3 wypełniona początkowymi losowymi wartościami:	Macierz po dokonaniu obróbki:
1 3 1 3 0 2 2 2 3 0 1 2	1 4 5 3 3 5 2 4 7 0 1 3

Blok warunkowy

```
for i=1:9
  if i <= 3
    i
  elseif (i <= 7)
    i+10
  else
    i+100
  end;
end; clear i
```

Zadanie. Proszę stworzyć macierz o rozmiarze 3x4 wypełnioną losowymi wartościami całkowitymi z przedziału [0; 5], a następnie proszę za pomocą podwójnej pętli i polecenia **if** wyzerować wartości większe niż 3. Na przykład:

Macierz 3x4 wypełniona początkowymi losowymi wartościami:	Macierz po dokonaniu obróbki:
2 3 0 0 4 3 4 1 4 2 3 1	2 3 0 0 0 3 0 1 0 2 3 1

Dynamiczna zmiana rozmiaru macierzy (bardzo wolna metoda)

```
bb = [];
bb = [bb 1:3]
bb(6) = 10
bb(end+1) = 11  %dodanie wartosci
```

```
bb = [bb; 101:107]

bb = 1:10;
bb(bb > 3 & bb < 8) = []; %nietypowy sposob usuwania elementow
```

Zmienna typu komórkowego

```
bb = {}        % zmienna typu ‘komorkowego’,
               % kazdy element może byc dowolnego typu
bb{1} = 1; bb{2} = [1:4]; bb{3}='napis';
               % kazdy element moze byc innej wielkosci i typu
bb{2}(3)
```

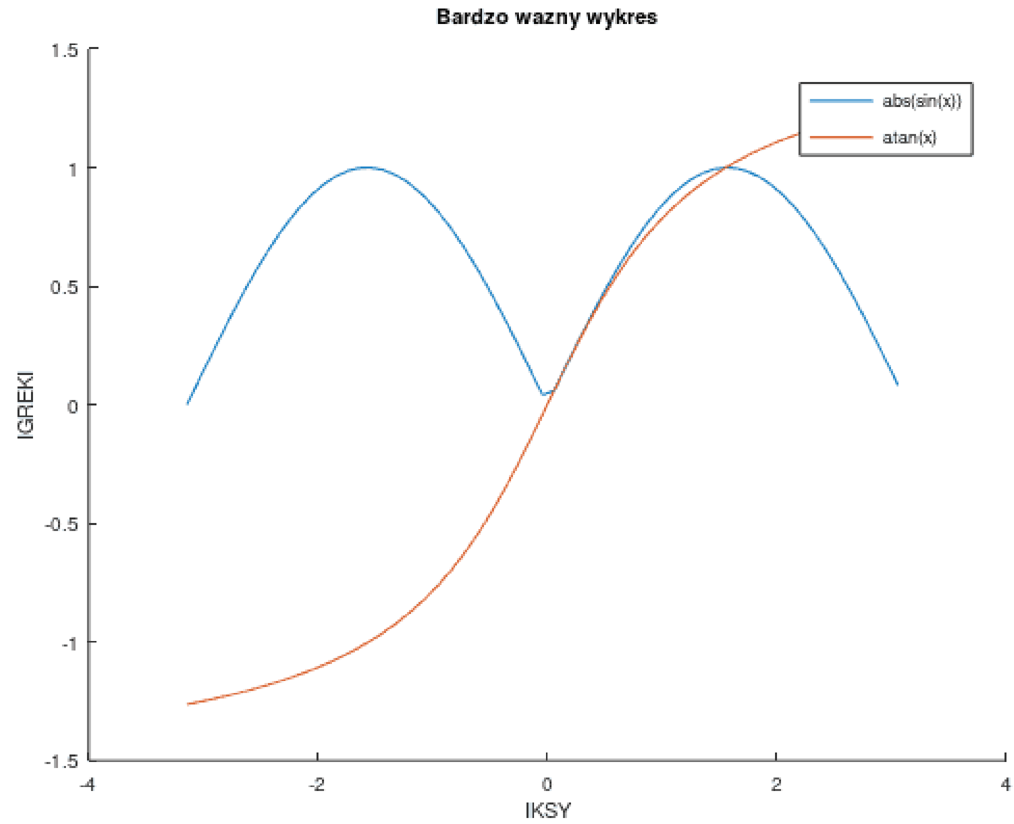
Typowe wykresy

```
plot(-10:10)
plot(-10:10, '.-r')      % czerwona („r”) linia
                         % przedzielona kropkami („.-”).

doc plot                 % Lista dostepnych opcji komendy plot
plot(-10:10, [101:121], ':k', 'linewidth', 10)
xlabel('wymiar x'); ylabel('wym y'); title('tytul');
xlabel('\alpha_{\Gamma}^{123}','Interpreter','Tex');
clf;      % czyszczenie wykresu

hold on;  % zablokowanie wykresu,
          % przygotowanie do kilku serii danych
plot(1:11, ':r'); plot(3:13, '--b'); plot(5:15, 'sk');
          % kilka serii danych na jednym wykresie
hold off;
legend('a1','b2', 'c3');
doc plot
xlim([0 10]); ylim([-10 10]);      % ograniczenie zakresu
                                   % wyswietlania wykresu
```

Wykresy 3d typu „dywanowego”

```
mac = [1 2 3; 3 2 1; 3 3 3];
mesh(mac)
meshc(mac)
surf(mac)
```

Zadanie. Proszę wyświetlić na jednym wykresie funkcje: abs(sin(x)) i atan(x) dla x mieszczącego się w przedziale (-pi; pi). Należy odpowiednio opisać osie OX i OY oraz dodać opis (legendę) dla obu funkcji oraz tytuł dla całego wykresu.



%%
%%

Podstawowe operacje na obrazkach

Odczyt obrazka

```
img_uint8 = imread('rzeczka.jpg');
     % zwroci dane w formacie uint8
     % typ ten jest bardzo kiepsko obslugiwany przez Matlaba/Octave
     % wiele operacji nie chce dzialac lub zwraca
     % niespodziewane wyniki.
     % Zaleca sie pracowac na liczbach typu double.
```

Wyświetlenie obrazka

```
imshow(img_uint8);
```

Zapis obrazka

```
imwrite(img_uint8, ‘wynik.png’);
```

Konwersja na wygodny dla Matlaba/Octave typ danych

```
img_dbl = double(img_uint8);
img_uint8 = uint8(img_dbl);        % konwersja w druga strone

img_dbl(3:4, end-3:end, :)         % prosze popatrzec na typ danych
```

Wyświetlanie obrazów dla typu danych double

```
imshow(img_dbl);    % blad, Octave przyjmuje, ze dla typu double
                    % uzyteczny zakres wartości to [0.0; 1.0]
                    % zamiast [0; 255]
imshow(img_dbl/255); imshow(uint8(img_dbl));% obie poprawne metody
```

Podobnie należy postąpić przy zapisie obrazka opisanego przez macierz typu double.

Zmiana zawartości pojedynczego piksela

```
img_dbl(end, 1, 1) = 0; img_dbl(end, 1, 2) = 255;
img_dbl(end, 1, 3) = 0;
     %zmiana koloru piksela w lewym dolnym rogu na zielony
```

Operacja agregujące na macierzach 3D.

Część operacji agregujących działa również na macierzach 3d, ale należy zwrócić uwagę na to, że podany wymiar będzie tym wymiarem, który ulegnie „zagregowaniu”, a pozostałe dwa wymiary pozostaną bez zmian, na przykład:

```
size(img_dbl)          % zwroci wektor [wysokosc szerokosc 3]
                       % dla obrazka typowego kolorowego
max(img_dbl, [], 2)    % zwroci macierz o wielkosci [wysokosc 1 3]
                       % czyli wymiar drugi (dot. szerokosci) zostanie
                       % zagregowany
```

Podział wykresu na kilka części

```
clf; % wyczyszczenie i przygotowanie miejsca na wykres
subplot(2, 2, 1);        % podzial wykresu na 2x2 czesci,
                         % skupienie sie na pierwszej cwiartce
                         % (lewa gorna cwiartka)
imshow(img_dbl(:, :, 1)/255);
     % wyswietlenie czerwonej skladowej obrazka
subplot(2, 2, 2);
imshow(img_dbl(:, :, 2)/255);
     % wyswietlenie zielonej skladowej obrazka
subplot(2, 2, 3);
imshow(img_dbl(:, :, 3)/255);
     % wyswietlenie niebieskiej skladowej obrazka
```

Zadania. związane z obróbką poniższego obrazu do wykonania

Figure 1: rzeczka.jpg

1. Proszę stworzyć macierz zawierającą obrazek „rzeczka.jpg”, następnie zmienić kolor kolumn pikseli od 10 do 50 na żółte, później wyświetlić tak powstały obrazek.

2. Proszę stworzyć macierz zawierającą po lewej stronie obrazek „rzeczka.jpg”, a po prawej jego lustrzane odbicie. Proszę nie używać *subplot* i *flip*.

3. Proszę stworzyć negatyw obrazka „rzeczka”, powinien on wyglądać tak, jak poniżej. Negatyw polega na zamianie wartości 0,1..255 na 255,254..0.