Zadanie 1

Podpunkt a)

Tworzymy

- Dwa wektory **a1** i **a2** o wymiarach **1x4**
- Macierz jednostkową **A1** wymiaru **2x2**
- Dowolną macierz górnotrójkątną **A2** wymiaru **2x2**

```
a1 = [1 2 3 4];
a2 = [1 1 1 1];
A1 = eye(2);
A2 = [1 2; 0 3];
```

Następnie łączymy wektory **a1**, **a2** oraz macierze **A1**, **A2** w jedną dużą macierz **M**.

```
M = [[a1; a2]; [A1 A2]]
```

```
M = 4×4
     1     2     3     4
     1     1     1     1
     1     0     1     2
     0     1     0     3
```

Wyliczamy wyznacznik oraz macierz odwrotną macierzy **M**

```
determinant = det(M)
```

```
determinant =
-8
```

```
inverse = inv(M)
```

```
inverse = 4×4
   -0.5000    1.0000    0.5000         0
         0    0.7500   -0.7500    0.2500
    0.5000   -0.5000         0   -0.5000
         0   -0.2500    0.2500    0.2500
```

Następnie zapisujemy do osobnych zmiennych

- Pierwszą kolumnę macierzy
- Trzeci wiersz macierzy
- Drugi i trzeci element czwartego wiersza macierzy
- Podmacierz od elementu **(2,2)** do elementu **(3,4)**

```
first_column = M(:,1)
```

```
first_column = 4×1
     1
     1
     1
     0
```

```
third_row = M(3,:)
```

```
third_row = 1×4
     1     0     1     2
```

```
second_and_third_element_of_fourth_row = M(4,2:3)
```

```
second_and_third_element_of_fourth_row = 1×2
     1     0
```

```
submatrix = M(2:3,2:4)
```

```
submatrix = 2×3
     1     1     1
     0     1     2
```

Podpunkt b)

Zapisujemy dane z poprzedniego podpunktu do pliku **matrices.mat**

```
save matrices.mat
```

Podpunkt c)

Czyścimy zmienne, następnie ładujemy dla testu plik **matrices.mat**

```
clear
```

```
load matrices.mat
```

Zadanie 2

Przywołujemy funkcję **validate(pesel)** z poprzednich zajęć.

```
function isValid = validate(pesel)
    pesel = num2str(pesel);

    if ~all(isstrprop(pesel, 'digit'))
        error('Dane wejściowe zawierają znaki nie będące cyframi');
    end

    if length(pesel) ~= 11
        error('PESEL musi mieć dokładnie 11 cyfr');
    end

    digits = arrayfun(@(x) str2double(x), pesel);
    weights = [1 3 7 9 1 3 7 9 1 3 1];
    weightedSum = sum(digits .* weights);

    isValid = mod(weightedSum, 10) == 0;
end
```

Wczytujemy dane z pliku **example-pesel-numbers.txt**. Następnie dla każdego przykładowego numeru w tym pliku tekstowym sprawdzamy czy jest poprawnym numerem PESEL czy nie, ponadto zapisujemy płeć.

```
% Read PESEL numbers from the text file
pesel_numbers = readlines('example-pesel-numbers.txt');

% Initialize empty vectors for storing results
pesels_amount = length(pesel_numbers);
```

```
validity = zeros(pesels_amount, 1);
sex = strings(pesels_amount, 1);

for i = 1:pesels_amount
    pesel = pesel_numbers(i);
    digits = arrayfun(@(x) str2double(x), num2str(pesel));

    isValid = validate(pesel);
    validity(i) = isValid;

    if isValid
        if mod(digits(10), 2) == 0
            sex(i) = 'Female';
        else
            sex(i) = 'Male';
        end
    else
        sex(i) = 'Null';
    end
end
```

Wyniki zapisujemy w tabeli **pesel_table** którą eksportujemy do pliku **pesel-table.xlsx**

```
pesel_table = table(pesel_numbers, validity, sex, 'VariableNames', {'PESEL',
'IsValid', 'Sex'});
writetable(pesel_table, 'pesel-table.xlsx');
disp(pesel_table);
```

PESEL	IsValid	Sex
"19220315474"	1	"Male"
"12345678903"	1	"Female"
"19900101234"	0	"Null"
"19750505678"	0	"Null"
"19880912345"	0	"Null"
"19660214789"	0	"Null"
"19520818901"	0	"Null"
"19430101234"	1	"Male"
"19310707890"	0	"Null"
"19220315478"	0	"Null"
"19130912345"	0	"Null"
"19040505678"	0	"Null"
"09040505678"	0	"Null"

Zadanie 3

Wczytujemy dane z pliku **stocks.csv** i dla każdego wiersza obliczamy różnice **close - open** oraz **max - min**. Wyniki zapisujemy w kolumnach **close_open_difference** i **max_min_difference**

```
table = readtable('stocks.csv');
table.close_open_difference = table.close - table.open;
table.max_min_difference = table.max - table.min;
table
```

table = 448×9 table

...

	name	date	open	max	min	close	volume
1	'06MAGNA'	20141201	2.3300	2.5000	2.3300	2.4500	7999
2	'08OCTAVA'	20141201	0.8600	0.8600	0.8600	0.8600	4230
3	'4FUNMEDIA'	20141201	6.4200	6.6800	6.4100	6.4100	5903
4	'ABCDATA'	20141201	3.5300	3.5300	3.4900	3.5000	48384
5	'ABPL'	20141201	32.1100	32.3500	31.9300	31.9300	1047
6	'ACAUTOGAZ'	20141201	28.8100	28.8100	28.8100	28.8100	4
7	'ACE'	20141201	10.6100	10.6100	10.5600	10.5900	2420
8	'ACTION'	20141201	44.8000	44.8000	43.3700	44.6500	3976
9	'AGORA'	20141201	8.1900	8.1900	8.0200	8.0800	22674
10	'AGROTON'	20141201	1.2000	1.2700	1.2000	1.2400	29124
11	'ALCHEMIA'	20141201	5.0200	5.0200	4.9700	5.0200	932486
12	'ALIOR'	20141201	78.5000	80.5000	78.5000	79.6500	154743
13	'ALMA'	20141201	16.3900	16.3900	15.5000	15.8000	401
14	'ALTA'	20141201	1.7900	1.8200	1.7000	1.8200	1684

⋮

Zadanie 4

Wczytujemy plik graficzny **`.jpg`** do macierzy **`M`**.

```
M = imread('rei.jpg');
```

Sprawdzamy wymiary tej macierzy.

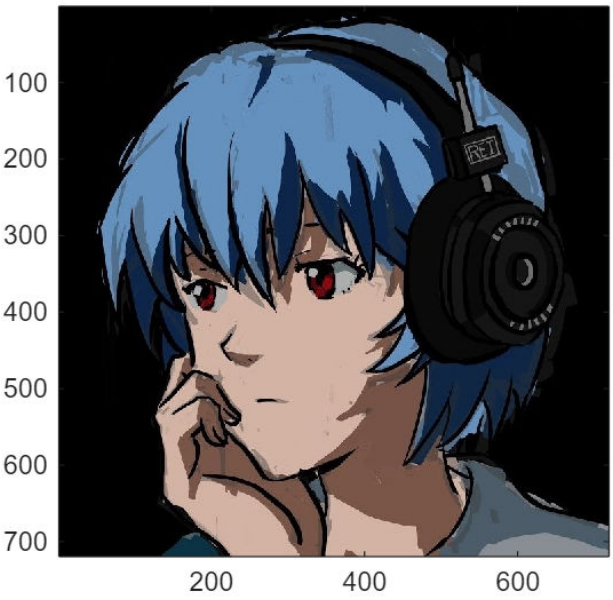
```
size(M)
```

```
ans = 1×3
   719   719     3
```

Pierwsze dwa wymiary odpowiadają rozdzielczości pliku graficznego, a pozostały 3 wymiar odpowiada kanałom kolorów czerwonego, zielonego i niebieskiego

Wyświetlamy obraz na podstawie macierzy **`M`**

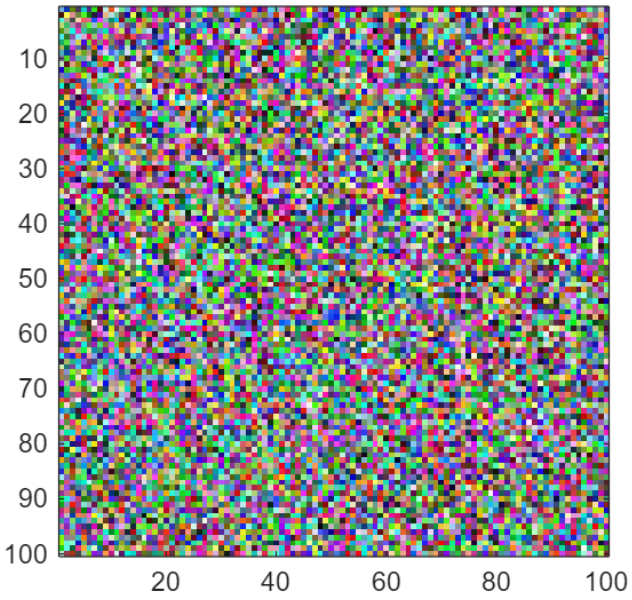
```
image(M)
axis image;
```



Eksperymentowanie z generowaniem obrazów

1. Losowy szum

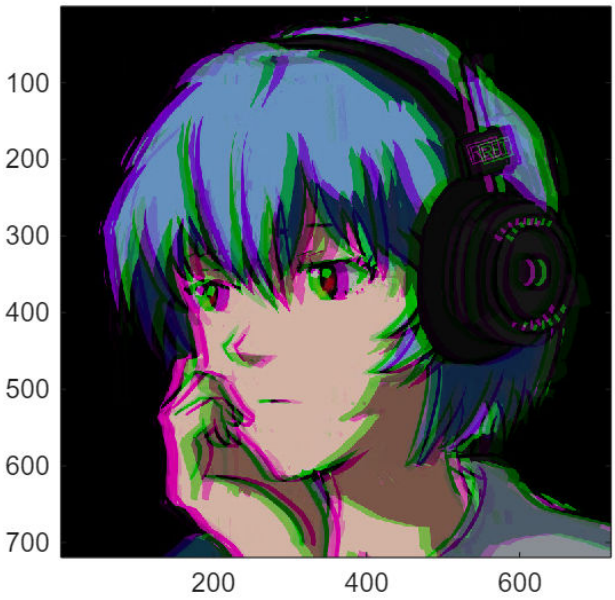
```
M = rand(100, 100, 3);
image(M);
axis image;
```



2. Efekt anaglifu

```
M = imread('rei.jpg');

[rows, cols, ~] = size(M);

green_shift = 15;
```

```
G = M(:,:,2);  % Green channel
G_shifted = G;  % Initialize G_shifted as G

% Shift the green channel
G_shifted(:, green_shift+1:end) = G(:, 1:end-green_shift);

M(:,:,2) = G_shifted;

image(M);
axis image;
```



3. Trójkąt Sierpińskiego

```
n = 512; % Size of the image
M = ones(n); % Empty matrix
p1 = [1, 1]; % Top vertex of the triangle
p2 = [n, 1]; % Bottom left vertex
p3 = [n/2, n]; % Bottom right vertex

% Initial random point within the triangle
point = [randi(n), randi(n)];

% Number of iterations
num_iter = 1000000;

% Generate the Sierpiński triangle
for i = 1:num_iter
    % Choose a random vertex
    choice = randi(3);
    if choice == 1
        point = (point + p1) / 2;
    elseif choice == 2
        point = (point + p2) / 2;
    else
        point = (point + p3) / 2;
    end
```

```
    % Mark the point in the matrix
    M(round(point(2)), round(point(1))) = 0;
end

% Convert M to a 3D matrix for display
Image = zeros(n, n, 3);
Image(:,:,1) = M;
Image(:,:,2) = M;
Image(:,:,3) = M;

% Display the Sierpiński triangle
image(Image);
axis image;
axis off;
```