Lab 3

Contents

1 Vectors warm-up

1.1 Array

An array, is a data structure consisting of a collection of elements (values or variables), each identified by at least one array index or key.

1.2 Vector

Vector is one-dimensional array.

1.3 Row vectors

```matlab
driverSalary = [1000, 2000, 3000, 4000];

driverSalary
driverSalary(1)
driverSalary(2)
driverSalary(end)
driverSalary(1:3)
driverSalary(:)

driverSalary(2) = driverSalary(2)-200;
driverSalary

length(driverSalary)
size(driverSalary)
```

1.4 Column vectors

Note semicolon instead of comma.

```matlab
driverSalary = [1000; 2000; 3000; 4000];

driverSalary
driverSalary(1)
driverSalary(end)
driverSalary(:)

length(driverSalary)
size(driverSalary)
```

1.5 Operations with vectors

```
% change all elements of a vector
driverSalary = driverSalary + 1000
driverSalary = driverSalary - 1000
driverSalary = driverSalary * 1000
driverSalary = driverSalary / 1000

array1 = [10, 20, 30];
array2 = [30, 20, 10];

% element-wise operations
array1 + array2
array1 - array2
array1 .* array2
array1 ./ array2

% vectors concatenation
[array1, array2]

% transpose
array1'

% concatenation again
[array1'; array2']
```

1.6 Other ways to create vectors

```
1:1:10
% or
1:10

1:2:10

-1:-1:-10

1:-1:10 % empty vector -- we cannot create a vector from 1 to 10
        % with step -1
```

```
linspace(1, 10, 5)

zeros(1, 10)
ones(1, 10)
rand(1, 10)

% and many other
```

1.7 Sum of all elements in a vector

```
clear;

vector = [1, 20, -3, 5, 6];
vector_length = length(vector);

sum = 0;

for i = 1:vector_length
    element = vector(i);
    sum = sum + element;
end
```

2 Matrices warm-up

Matrices are two-dimensional arrays. Each value in a matrix is identified by a pair of numbers: row and column.

2.1 Working with matrices

Matlab syntax for an element at row `r` and column `c` of the matrix `M` is `M(r, c)`.

Examples:

```
a = [10, 20.11; 3.18, pi];

a
a(1, 1) % element at the first row and first column
a(2, 1) % element at the second row and first column
```

```
a(end, 1) % element at the last row and first column
a(1:2, 1) % first and second rows of the matrix a and the first column
a(1, :) % first row of the matrix a
a(:, 2) % second column of the matrix a

a(2, 2) = -2; % changing value at the second row and second column
              % to -2
a(2, :) = a(2, :) + 3; % add 3 to all elements in the second row
a(:, 1) = a(:, 1) - 1; % add -1 to all elements in the first column
a

size(a) % returns size of the matrix
size(a, 1) % returns the number of rows in the matrix
size(a, 2) % returns the number of columns in the matrix
length(a) % returns the maximum of the number of columns and the number of rows
numel(a) % returns the number of elements in the matrix
```

2.2 Matrix operations

Very similar to operations with vectors.

```
% change all elements of the matrix a
a = a + 10
a = a - 10
a = a * 10
a = a / 10

matrix1 = [10, 20; 30, 40];
matrix2 = [30, 20; 10, 66];

% element-wise operations
matrix1 + matrix2
matrix1 - matrix2
```

```
matrix1 .* matrix2
matrix1 ./ matrix2

% matrix concatenation
[matrix1, matrix2]
[matrix1; matrix2]

% adding a column
matrix = [1, 2; 1, 2];
column = [3; 3];
matrix = [matrix, column];
% or
matrix = [matrix column];

% adding a row
matrix = [1, 1; 2, 2];
row = [3, 3];
matrix = [matrix; row];

% removing a row
matrix = [1, 1, 1; 2, 2, 2; 3, 3, 3];
matrix(3, :) = [];

% removing a column
matrix = [1, 2, 3; 1, 2, 3; 1, 2, 3];
matrix(:, 3) = [];

% transpose
a'
a''
```

2.3 Matrix creation

```
zeros(5, 10)
ones(6, 10)
rand(7, 10)
```

2.4 Iterating through all elements of the matrix

In order to work with elements of the matrix one-by-one, we need to use nested loops.

2.4.1 Print each element separately

In the following example we print each element separately:

```
clear;

M = rand(4, 4);

number_of_rows = size(M, 1);
number_of_columns = size(M, 2);

% for each row
for row = 1:number_of_rows
    % for each column
    for column = 1:number_of_columns
        disp(M(row, column)); % print value
    end
end
```

2.4.2 Adding 2 to each element

In the following example we add 2 to each element:

```
clear;

M = zeros(4, 4);

number_of_rows = size(M, 1);
number_of_columns = size(M, 2);

% for each row
for row = 1:number_of_rows
    % for each column
    for column = 1:number_of_columns
        M(row, column) = M(row, column) + 2; % add 2
    end
```

```
end
```

2.4.3 Finding the sum of all elements

In the following example we are calculating the sum of all elements in the matrix:

```
clear;

M = [1, 2, 1; 4, 5, 2; 1, 3, 2];
sum_of_elements = 0;

number_of_rows = size(M, 1);
number_of_columns = size(M, 2);

% for each row
for row = 1:number_of_rows
    % for each column
    for column = 1:number_of_columns
        sum_of_elements = sum_of_elements + M(row, column);
    end
end

disp(sum_of_elements);
```

2.4.4 Finding max element

In the following example we are looking for the largest element:

```
clear;

M = [1, 2, 1; 4, 5, 2; 1, 3, 2];
maximum = M(1, 1);

number_of_rows = size(M, 1);
number_of_columns = size(M, 2);

% for each row
for row = 1:number_of_rows
```

```
    % for each column
    for column = 1:number_of_columns
        if M(row, column) > maximum
            maximum = M(row, column);
        end
    end
end

disp(maximum);
```

2.4.5 Print matrix line by line

In the following example we are printing the matrix one row at a time:

```
clear;

M = rand(4, 4) ;
number_of_rows = size(M, 1) ;

% for each row
for row = 1:number_of_rows
    disp(M(row, :)) ; % print value
end
```

3 Task 3.1

Write a program for a lunch restaurant owner. The program should assist with computing the price of lunch for several people. Price of a lunch depends on the age of the visitor. If the visitor is over 10 years old then the lunch costs 110 SEK. Children younger than 10 eat for free.

The program should:

1. Ask the user to enter the number of visitors.

2. Creates a matrix of the corresponding size where ages of visitors and the prices will be stored.

3. Request the age of every visitor and compute the cost of lunch for this visitor.

4. Ages of visitors entered by the user and the corresponding prices should be stored in a matrix.

5. After all visitors' information has been entered, the program should form the bill, that is, print the matrix with ages and prices for all visitors and print the total price of lunch for all visitors.

Example of the program:

```
Enter the number of visitors: 5
Enter the age of visitor 1: 10
Enter the age of visitor 2: 25
Enter the age of visitor 3: 8
Enter the age of visitor 4: 11
Enter the age of visitor 5: 65

Visitor age and price
    10     0
    25   110
     8     0
    11   110
    65   110

==============================
Total price: 330
```

4 Task 3.2

Write a script which asks the dimension of the vector from the user, creates two vectors and fills them with random numbers between -100 and 100. Then it prints both vectors to the screen and does the following:

- Prints one by one all elements of the second vector which are less than the element of the first vector at the same index.

- If there are no such elements prints the corresponding message to the screen

Do not use vectorization techniques, use loops instead.

For example, for vectors $a = [10, -10, 20, 30]$ and $b = [20, -20, 10, 40]$ the program should print

```
-20
10
```

For vectors $a = [10, -10, 20, 30]$ and $b = [20, 0, 30, 40]$ the program should print "No such elements" (or a similar message).

5 Task 3.3

Write a script that asks the dimension of the matrix from the user, fills it with random numbers between -100 and 100 and prints the following:

- The product of all negative elements in the column (for each column)
- The minimum element of the matrix and the row with this element

Print the matrix itself also.
Do not use `min()` or `prod()`, do it using for loops.