Lab #3

Contents

1 For loops continuation

Consider the following small algorithm:

```
k = 0
for n=1:5
    k=k+1
end
```

What is the value of `k` on each step?
Try to do the same with the following algorithm:

```
k = 1
for n=1:5
    k=k*n
end
```

2 Conditional statements

2.1 Boolean data type

Boolean data type takes one of two possible logical values: "true" or "false". Matlab stores "false" and "true" as 0 and 1 respectively.

2.2 Conditional expressions

The result of a conditional expression is of boolean type.

Examples of conditional expressions

```
a = 10;

a < 30 % true
a > 20 % false
a == 10 % true
a == 11 % false
```

2.3 if statement

`if` statement allows you to perform different computations or actions depending on whether a condition evaluates to true of false.

Basic syntax is:

```
if condition
    code
end
```

Example:

```
a = 10; % try to change the value!

if a > 9
    disp('a is greater than 9');
```

```
end


if a < 20
    disp('a is less than 20');
end
```

2.4 else statement

Code in the `else` block will be executed if `condition` is false.

Basic syntax:

```
if condition
    code1
else
    code2
end
```

Example:

```
a = 20;

if a < 40
    disp('a is less than 40');
else
    disp('a is greater than 40');
end
```

2.5 elseif statement

`elseif` statement allows to combine several conditions. Only the code following the first condition that is found to be true will be executed. All other code will be skipped.

Basic syntax:

```
if condition1
    code1
elseif condition2
    code2
elseif condition3
    code3
```

```
...
else
    code
end
```

Examples:

```
a = 10; % try to set a = 4; a = 5; a = 6;

if a > 5
    disp('a > 5');
elseif a < 5
    disp('a < 5');
else
    disp('a == 5');
end
```

2.6 and/or/not operators

If you want to have complex conditions which consist of more than one logical statement, you can use logical "and", "or" and "not" operators.

2.6.1 And

The "and" of two or more conditions is true if each of the conditions is true. For example, `a and b` is true only if `a` and `b` are both true.

In Matlab, logical "and" is written as `&&`.

Example:

```
a = 10;

if a > 5 && a < 15
    disp('a > 5 and a < 15');
else
    disp('a <= 5 or a >= 15');
end
```

2.6.2 Or

The "or" of two or more conditions is true if at least one of the conditions is true. For example, `a or b` is true if either `a` or `b` (or both) are true.

In Matlab, logical "or" is written as `||`.

Example:

```
a = 10;

if a < 5 || a > 9
    disp('a < 5 or a > 9');
else
    disp('9 => a >= 5');
end
```

2.6.3 Not

`not` operator negates the condition. If `a` is true, then `not a` is false. If `a` is false, then `not a` is true.

In Matlab, logical "not" is written as `~`.

Example:

```
a = 10

if ~(a > 0)
    disp('a <= 0');
else
    disp('a > 0');
end
```

3 Task 5

Given a vector `v = [10, -2, -331, 1, -100, 201]`

- create a new vector `v_positive` containing only positive elements of the vector `v`. The result should be `[10, 1, 201]`.
- create a new vector `v_negative` with the negative elements of `v`. The result should be `[-2, -331, -100]`.
- construct a vector `v_concat` as a concatenation of `v_positive` and `v_negative`. The result should be `[10, 1, 201, -2, -331, -100]`.
- print to output all 3 vectors and their sizes.

Hint: to create an empty vector `v_positive` you can use the following code

```
v_positive = [];
```

And you can later append elements to it by using **concatenation**

```
v_positive = [v_positive, new_element]; % to add a
    new element "new_element" to the vector
```

4 Task 6

Modify task 3 to compute the product of only **negative** numbers of the vector. For example, if the input vector `v = [-3, 100, -5, -6, 20, 7]` the result should be -90.