Review of Lecture 7

- **`if-else` statement**

```
if   condition
 <Instructions>
end
```

```
if  condition
 <Instruction 1>
else
 <Instruction 2>
end
```

```
if condition 1
   <Instructions 1>
elseif condition 2
   <Instructions 2>
else
   <Instructions 3>
end
```

- **`Switch` statement**

```
switch expression
case caseValue1
        < code 1 >
case caseValue2
        < code 2 >
    ...
otherwise
         < code n >
end
```

Info 3

Introduction to MATLAB

M. Bouzenita
2nd year Engineer - University of Jijel

Lecture 8

Control instructions (for and while loops)

1. Control instructions for and while loops

In MATLAB, we can distinguish two types of loop statements :

- **Counted loop** in which the statements are repeated with a specified number of times.
- **Conditional loop** where the statements are executed until reaching defined goal or conditions become false.

The **counted loop** is based on the `for` statement and the `while` statement is usually adopted for **conditional loop**.

2. `for` loop

In general, we use the `for` loop to repeat a set of statements with a known ahead times called actions of the loop.

```
for iterate_var = range
    <action>
end
```

where

- `iterate_var` represents the loop variable,
- `range` indicates the range in which the loop variable iterates and
- `<action>` represents the set of statements to be repeated until completing all iterations.

2. `for` loop

In general, we use the **`for`** loop to repeat a set of statements with a known ahead times called actions of the loop.

```
for iterate_var = range
    <action>
end
```

Printing three times 'Hello world'

for_loop_msg.m

```
for i = 1:3

    disp('Hello world');

end
```

```
>> for_loop_msg
Hello world
Hello world
Hello world
```

2. `for` loop

In general, we use the **`for`** loop to repeat a set of statements with a known ahead times called actions of the loop.

```
for iterate_var = range
    <action>
end
```

calculate the square of the first three numbers in which the loop variable is used.

for_loop.m

```
for i = 1:3

    s = i^2;

    fprintf('The square of %d is %d \n',i,s)
end
```

```
>> for_loop
The square of 1 is 1
The square of 2 is 4
The square of 3 is 9
```

2. `while` loop

Instead of repeating statements with a known ahead times, the `while` loop **repeats the actions as long as the condition is true**.

```
while condition
    <action>
end
```

where the condition is evaluated in each iteration and if it is logically **true**, the action is executed until the condition becomes **false**.

To avoid an **infinite** loop in which the execution of the action never halts, **the condition must become false**.
In MATLAB, if the infinite loop occurs, we press **Ctrl-C** to exit the loop.

2. `while` loop

Instead of repeating statements with a known ahead times, the **`while`** loop **repeats the actions as long as the condition is true**.

```
while condition
    <action>
end
```

Printing four times 'Hello world'

while_msg.m

```
itr = 1

while itr < 5

    disp('Hello world');

itr = itr + 1;

end
```

```
>> while_msg
Hello world
Hello world
Hello world
Hello world
```

2. `while` loop

Instead of repeating statements with a known ahead times, the **`while`** loop **repeats the actions as long as the condition is true**.

```
while condition
    <action>
end
```

Prompt the user to input appositive number.

Pos_num.m

```
n = -1;
while n <= 0
 n = input('Please enter a positive number:');
end
fprintf('You entered : %s\n', num2str(n));
```

```
>> Pos_num
Please enter a positive number:  -5
Please enter a positive number:  -7
Please enter a positive number:  -0.256
Please enter a positive number:   5.3
You entered : 5.3
>>
```

2. `while` loop

Instead of repeating statements with a known ahead times, the **`while`** loop **repeats the actions as long as the condition is true**.

```
while condition
    <action>
end
```

calculation of the factorial of a number
Exp: 5! = 5 x 4 x 3 x 2 x 1
20 x 3 x 2 x 1
60 x 2 x 1
120 x 1
120

factorial.m

```
n = input('Please enter a positive number: ');
my_n = n; % Save n value to use it in fprintf
f = n;
while n > 1
    n = n-1;
    f = f*n;
end
fprintf('%d! = %d \n',my_n, f)
```

```
>> factorial
Please enter a positive number: 8
8! = 40320
```

2. `while` loop

Instead of repeating statements with a known ahead times, the **`while`** loop **repeats the actions as long as the condition is true**.

```
while condition
    <action>
end
```

In some cases, we need to quit a **`for`** or a **`while`** loop before reaching the end, then we use the **`break`** statement to exit the loop.
However, if we need to skip the rest of actions in the loop and begin the next iteration, we use the **`continue`** statement.
In MATLAB, **`break`** is defined only inside a for or a while loop.

Practice