MAT 1206 – Introduction to MATLAB

CHAPTER 03: Basic programming structures

Lesson 2: Loops, Nested loops, Loop control statements

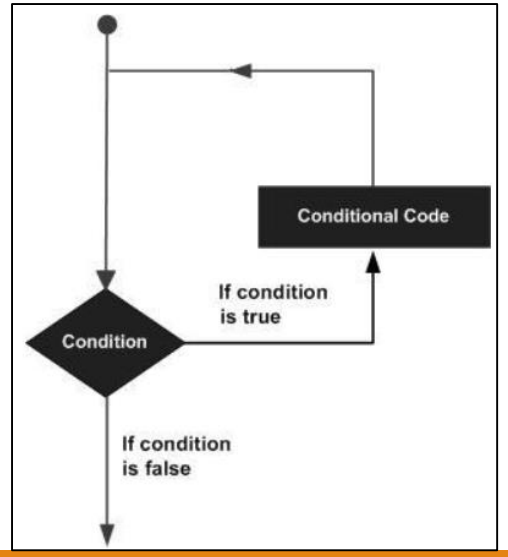
Content

- While loop
- For loop
- Nested loop
- Loop control statements
  - Break
  - continue

Loop Types

There may be a situation when you need to execute a block of code several number of times. In general, statements are executed sequentially. The first statement in a function is executed first, followed by the second, and so on.

A loop statement allows us to execute a statement or group of statements multiple times and following is the general form of a loop statement in most of the programming languages:



Cont.

MATLAB provides following types of loops to handle looping requirements.

Loop Type	**Description**
While loop	Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.
For loop	Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable.
Nested loops	You can use one or more loops inside any another loop.

The while Loop

Repeats a statement or group of statements while a given condition is true. It tests the condition before executing the loop body.

Syntax:

```
while <expression>

   <statements>

end
```

Example:

```
a = 10;
% while loop execution
while( a < 20 )
      disp(['value of a: ', num2str(a)]);
      a = a + 1;
end
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

The for Loop

A for loop is a repetition control structure that allows you to efficiently write a loop that needs to execute a specific number of times.

Syntax:

```
for index = values
  <program statements>
        ...
end
```

values has one of the following forms:

Format	Description
initval:endval	increments the index variable from initval to endval by 1, and repeats execution of program statements until index is greater than endval.
initval:step:endval	increments index by the value step on each iteration, or decrements when step is negative.
valArray	creates a column vector index from subsequent columns of arrayvalArray on each iteration.

Cont.

Example: 1

```
for a = 10:20
        disp(['value of a: ', num2str(a)]);
end
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
value of a: 16
value of a: 17
value of a: 18
value of a: 19
value of a: 20
```

Example: 2

```
for a = 1.0: -0.1: 0.0
        disp(a)
end
```

```
 1
0.9000
0.8000
0.7000
0.6000
0.5000
0.4000
0.3000
0.2000
0.1000
 0
```

Cont.

Example: 3

```
for a = [24,18,17,23,28]
        disp(a)
end
```

```
24
18
17
23
28
```

The Nested Loops

MATLAB allows to use one loop inside another loop.

Syntax:

```
for m = 1:j
    for n = 1:k
        <statements>;
    end
end
```

```
while <expression1>
   while <expression2>
      <statements>
   end
end
```

Cont.

Example:

```
for i = 1:3
        for j = 1:4
                disp([i, j]);
        end
end
```

```
Loops
     1     1
     1     2
     1     3
     1     4
     2     1
     2     2
     2     3
     2     4
     3     1
     3     2
     3     3
     3     4
```

```
i=1;
while i<4
        j=1;
        while j<5
                disp([i,j])
                j=j+1;
        end
        i=i+1;
end
```

fprintf

In MATLAB, **fprintf** is a function used for formatted printing of text and data to the command window or to a file. It allows you to control the output format and specify how variables are displayed.

Syntax: **fprintf(format, arg1, arg2, ...)**

Here, **format** is a string that specifies the format of the output, and **arg1**, **arg2**, etc., are the values or variables to be displayed according to the specified format.

The **format** string can include format specifiers, such as **%d** for integers, **%f** for floating-point numbers, **%s** for strings, and so on. These specifiers are used to define the type and formatting of the corresponding arguments.

Cont.

Example:

```
x = 10;
y = 3.14159;
str = 'Hello, MATLAB!';

fprintf('Integer: %d\n', x);
fprintf('Floating-point: %.2f\n', y);
fprintf('String: %s\n', str);
```

```
Integer: 10
Floating-point: 3.14
String: Hello, MATLAB!
```

In this example, we use **fprintf** to display the values of x, y, and str according to the specified formats. The **%d** specifier is used for the integer, **%.2f** specifies the floating-point number with two decimal places, and **%s** is used for the string.

Cont.

Example:

```
for i = 1:5
        for j = 1:i
                fprintf('%d ', j);
        end
        fprintf('\n');
end
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Cont.

Example:

```
num1 = 1;
while num1 <= 5
        num2 = 1;
        while num2 <= num1
                fprintf('%d ', num1 * num2);
                num2 = num2 + 1;
        end
        fprintf('\n');
        num1 = num1 + 1;
end
```

```
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25
```

Cont.

Example:

```
for i = 1:5
        for j = 1:i
                fprintf('%d ', i);
        end
        fprintf('\n');
end
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

Cont.

Example:

```
for i = 5:-1:1
        for j = 1:i
                fprintf('%d ', i);
        end
        fprintf('\n');
end
```

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

```
for i = 5:-1:1
        for j = 1:i
                fprintf('%d ', i);
        end
        fprintf('\n');
end
```

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Loop Control Statements

Loop control statements change execution from its normal sequence. When execution leaves a scope, all automatic objects that were created in that scope are destroyed.

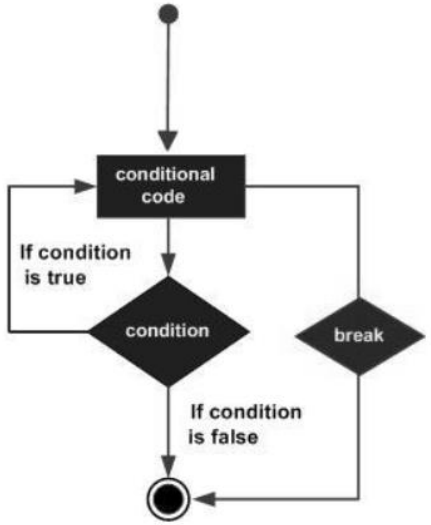
MATLAB supports the following control statements

Control statement	Description
break statement	Terminates the loop statement and transfers execution to the statement immediately following the loop.
continue statement	Causes the loop to skip the remainder of its body and immediately retest its condition prior to reiterating.

The break Statement

The break statement terminates execution of for or while loop. Statements in the loop that appear after the break statement are not executed.

In nested loops, break exits only from the loop in which it occurs. Control passes to the statement following the end of that loop.



Cont.

Example:

```
a = 10;
% while loop execution
while (a < 20 )
      fprintf('value of a: %d\n', a);
      a = a+1;
      if( a == 16)
            % terminate the loop using break statement
            break;
      end
end
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 15
```

Cont.

Example:

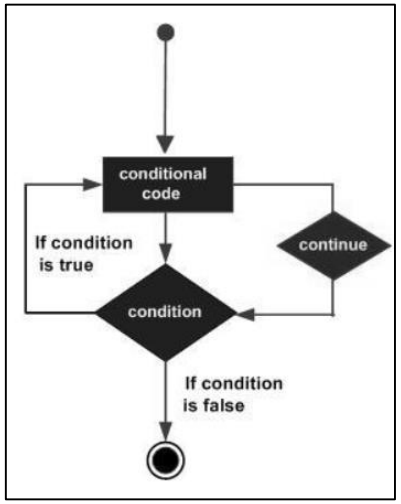
```
outerLimit = 4;
innerLimit = 3;
% Iterate through the outer loop
for i = 1:outerLimit
    % Iterate through the inner loop
    for j = 1:innerLimit
        disp(['Inner loop iteration (i,j): ' num2str([i,j])]);
        % Check a condition
        if i == 3 && j == 2
            disp('Breaking out of the nested loop...');
            break; % Exit the inner loop when condition is met
        end
    end
end
```

```
Inner loop iteration (i,j): 1  1
Inner loop iteration (i,j): 1  2
Inner loop iteration (i,j): 1  3
Inner loop iteration (i,j): 2  1
Inner loop iteration (i,j): 2  2
Inner loop iteration (i,j): 2  3
Inner loop iteration (i,j): 3  1
Inner loop iteration (i,j): 3  2
Breaking out of the nested loop...
Inner loop iteration (i,j): 4  1
Inner loop iteration (i,j): 4  2
Inner loop iteration (i,j): 4  3
```

The continue Statement

The **continue** statement is used for passing control to next iteration of for or while loop.

The **continue** statement in MATLAB works somewhat like the break statement. Instead of forcing termination, however, 'continue' forces the next iteration of the loop to take place, skipping any code in between.



Cont.

Example:

```
a = 10;
%while loop execution
while a < 20
      if a == 15
            % skip the iteration
            a = a + 1;
            continue;
      end
      fprintf('value of a: %d\n', a);
      a = a + 1;
end
```

```
value of a: 10
value of a: 11
value of a: 12
value of a: 13
value of a: 14
value of a: 16
value of a: 17
value of a: 18
value of a: 19
```

Questions/queries?